DESIGN AND IMPLEMENTATION OF AN ATTRIBUTE BASED AUTHORIZATION MANAGEMENT SYSTEM

A Thesis
Presented to
The Academic Faculty

by

Apurva Mohan

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
May 2011

DESIGN AND IMPLEMENTATION OF AN ATTRIBUTE BASED AUTHORIZATION MANAGEMENT SYSTEM

Approved by:

Professor Douglas Blough,
Committee Chair
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Douglas Blough, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor George Riley
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Mustaque Ahamad
College of Computing
Georgia Institute of Technology

Professor Ling Liu
College of Computing
Georgia Institute of Technology

Professor Chuanyi Ji
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Daniel Russler
Health Sciences Global Business Unit
Oracle Corporation

Date Approved: 04-01-2011

Dedicated to my parents, Mrs. Arti Mohan and Dr. Surendra Mohan

and

my beloved wife, Himali

Without their support at every stage, this PhD would have remained a dream.

ACKNOWLEDGEMENTS

Neither the miles would have been reached nor the mission accomplished, without the help of those who have directly and indirectly helped me in completing this research. First and the foremost, I would like to thank my advisor, Dr. Douglas Blough for his continuous encouragement, advice, and support. His support for my several activities in addition to research gave me an opportunity to gain a wider perspective of the field. I would also like to thank the members of my thesis committee for their thoughtful comments and valuable advice.

I would like to thank our research collaborators and faculty members who provided encouragement during my time at graduate school. I would especially like to thank Dr. A. P. Sakis Meliopoulos, Dr. Stylianos Kavadias, Dr. John Copeland, and Dr. Daniel Russler for helping and advising me at different stages.

I would like to take this opportunity to thank Margi Berbari and members of my TI:GER team for their association and knowledge sharing. Exploring the opportunities for commercializing the technology developed as part of my PhD research during this program helped me in understanding the bigger picture and also improved my appreciation of business and legal issues related to technology commercialization.

I would like to thank my classmates in graduate school and my labmates in the critical networking lab for their friendship and association. In addition, I was fortunate to have a number of good friends inside and outside Georgia Tech and I will always cherish having them as friends.

I consider myself extremely fortunate to be part of a family which greatly values education which motivated me to pursue graduate studies. I would especially like to thank my parents and siblings Amit, Roli, and Ritu for their strong encouragement

that helped me in fighting the odds. Last but not the least, I would like to thank my wife Himali who have supported me in all my decisions and have been a constant pillar of strength. Our first few years together as graduate students and sharing the times through thick and thin have created some cherished memories.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The proposed research is in the area of attribute-based authorization systems. We address two specific research problems in this area. First, evaluating authorization policies in multi-authority systems where there are multiple stakeholders in the disclosure of sensitive data. The research proposes to consider all the relevant policies related to authorization in real time upon the receipt of an access request and to resolve any differences that these individual policies may have in authorization. Second, to enable a lot of entities to participate in the authorization process by asserting attributes on behalf of the principal accessing resources. Since it is required that these asserted attributes be trusted by the authorization system, it is necessary that these entities are themselves trusted by the authorization system. Two frameworks are proposed to address these issues. In the first contribution a dynamic authorization system is proposed which provides conflict detection and resolution among applicable policies in a multi-authority system. The authorization system is dynamic in nature and considers the context of an access request to adapt its policy selection, execution and conflict handling based on the access environment. Efficient indexing techniques are used to increase the speed of authorization policy loading and evaluation. In the second contribution, we propose a framework for service providers to evaluate trust in entities asserting on behalf of service users in real time upon receipt of an access request. This trust evaluation is done based on a reputation system model, which is designed to protect itself against known attacks on reputation systems.

CHAPTER I

INTRODUCTION

1.1 Authorization systems

In computer security, authorization is the function of specifying and enforcing access rights that a principal has on a resource. A principal is any entity, human or computer, that tries to access a resource in a system. Authorization is a mandatory part of modern computer security systems and a complementary function to authentication. While authentication answers the question *who is this principal?*, authorization answers the question *what is this principal allowed to access in the system?*. In early systems, authentication and authorization were coupled together where a principal who is allowed to use a computer system could access any resource on the system. As the number of users became large, a need was felt to control access to sensitive documents and allow each user to access only those resources that were required to perform her job function. Each user's identity was mapped to a set of resources that she was allowed to access. These systems are generally referred to as identity based systems. As identity based systems became more sophisticated they evolved into two primary types - a) *principal-centric* asserting 'which resources can this principal access?' and b) *resource-centric* asserting 'which principals can access this particular resource'? The former are referred to as capability-based systems whereas the latter are referred to as access control lists (ACL). ACLs became the predominant access control model for identity based systems. With the deployment of large IT systems in enterprises, authorization became a prominent security tool to control access to digital resources. In enterprises, principals were employees of the organization who needed access to resources to perform certain duties related to their work. Since a large number of

employees needed access to the same kinds of resources, a lot of different resources had access rules with the same employees repeated in each rule. Also, management of access control rules was becoming a challenge because when a new employee was added or when an existing employee was removed from the system, each relevant access control rule was modified in the system. To make the authorization more aligned with the workflow, role based access control (RBAC) was proposed [76] [77]. In RBAC, access rights are given to user groups which are referred to as roles. Employees in an organization become members of these user groups based on the duty they perform in the organization and are said to hold those roles. Employees have to activate a specific role in order to use the privileges associated with that role.

Attribute-based authorization is a recent paradigm in authorization systems [86]. These systems provide fine granularity, high flexibility, rich semantics and other useful features like partial authentication and natural support for role-based access control. They are also very generic systems and are backward compatible with other technologies like role-based systems or identity-based systems. They can be used in a constrained fashion to achieve this backward compatibility. For the purposes of authorization, the appropriate authority specifies some authorization policies defining the attributes with some specific values (or range of values) that the principal needs to provide in a verifiable form. The attributes are in a verifiable form when they are asserted by some trusted entity on behalf of the user. The current research uses attribute-based authorization technology to provide secure authorization for protecting sensitive data.

1.2 Architecture of an attribute-based authorization system

In computer systems, users try to connect to other computers in order to acquire some service or data. These users are called data or service users and the computers which provide them service or data are generally known as Service Providers (SP).

Some SPs provide service which is open to all without any charge, while others either provide premium service or share sensitive data. It is imperative to define a set of users who can access service provided by SPs of the latter kind. For premium services, this set comprises users who have paid for the services and for sharing sensitive data the set comprises users who have been authorized to access this data. Authorization systems act as proxies to these service providers and check whether the users have authorization to receive the requested services. Some common types of authorization systems are discussed in Chapter 2. In all these systems, the basic authorization function remains the same whereas the methods to enforce authorization vary a lot from one technology to another.

In attribute-based systems, authorization is based on certain attributes which are authorization related. Modern systems use up to four types of attributes, which may be based on the characteristics of the user, the protected resource, the access environment or the actions that the user intends to perform on the desired resource.[1] A representative architecture of this system is given in Figure 1. In attribute-based systems, authorization permissions are mapped to user attributes (with specific values). A user has to prove that he holds these attributes (with specified values) to be authorized to access the desired resources. For our purposes, the authorization system is central to this architecture. It has three main modules viz. a repository of attribute-based authorization policies, credential verification and access control enforcement. All the relevant stakeholders in providing the service at the SP compose authorization policies based on user, resource, action and environmental attributes. Each stakeholder is commonly referred to as an 'authority' in the system. If there are multiple authorities specifying authorization policies in a system, it is referred to as a 'multi-authority system'. These policies are stored in a local repository and

[1]From the point of view of authorization, the service which a SP provides is considered a resource.

are fetched and enforced during the authorization process. The user presents his attributes asserted by some attribute providers (AP) by including the attributes and their values in digitally signed credentials. The credential verification module of the authorization system verifies these signed credentials and determines whether they are trusted attributes or not. Only attributes which are trusted by the authorization system can be used for authorization. The access control enforcement module uses the other two modules to determine whether to provide access to the service or not. It is composed of an authorization engine which uses verified attributes and compares them to the stored policies. Based on this comparison, it reaches a unique authorization decision to permit or deny access to the requesting user. It then enforces this decision by either sending a 'deny' to the user or by connecting him to the requested service provider. One variation of this model called 'partial permit' performs partial authorization where the requester is allowed to see a subset of the resources that he requested. In case of 'partial permit', subsets or groups of data have authorization rules. When the requester requests access to multiple groups of data, the authorization system verifies it against the policies and returns the groups which the requester is eligible to see. A typical case of 'partial permit' is a database query where the requester is allowed to see only some requested data elements in the database and not others. The authorization system determines the data set which the requesters can view and returns partial results for the query. Particular care has to be taken to convey this information to the requester so that he is aware that the returned result is a partial result. Failure to do so may lead to wrong analysis or interpretations, especially in case of statistical studies.

A unique feature of our approach is that we combine the use of quasi-static attributes like the role of the user or the name of the user's employer with highly dynamic attributes such as the user's location, time, or the characteristics associated

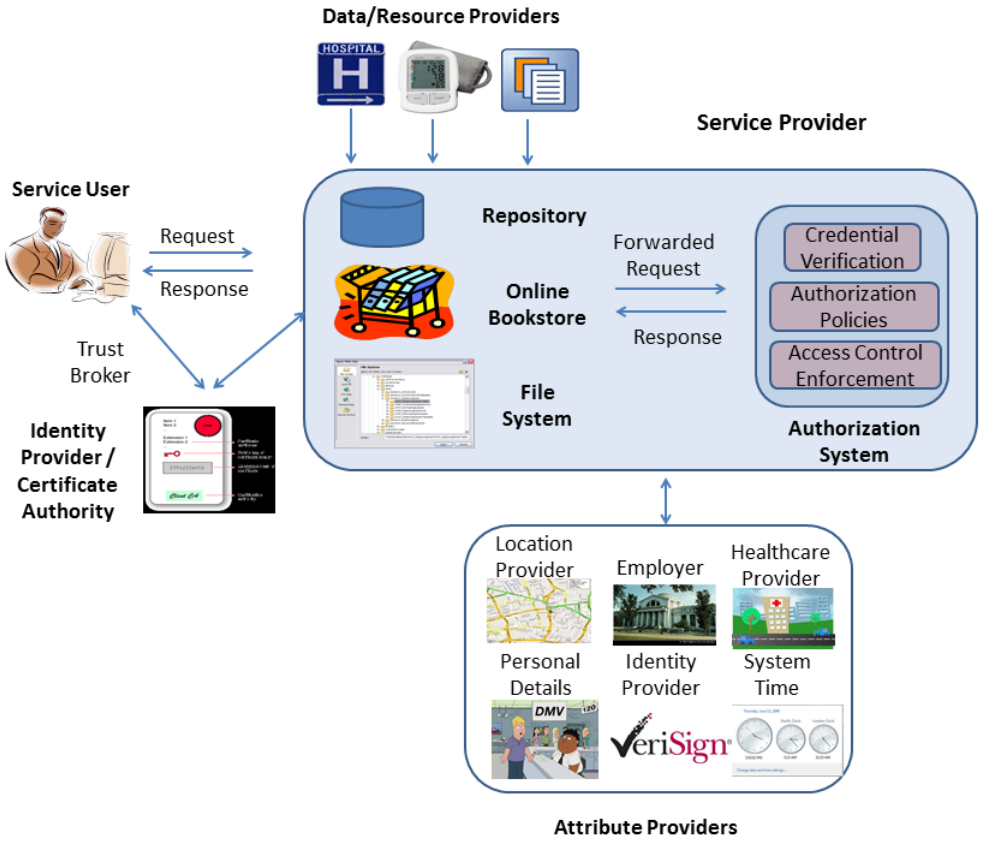

Figure 1: System representation of the attribute-based authorization system.

with an emergency situation (see Sections 5.3.1 and 5.4.2 for an example of this). Attribute providers (refer Figure 1) are entities that verify users' attributes and certify them. They create digitally signed credentials and provide them to the user. Our definition of an AP differs from that of an identity provider (IdP) in that an IdP only certifies identity related attributes, and a user usually has a small number of IdPs. On the other hand, there could be many APs providing attributes about a given user, and these attributes may or may not be identity related. A host of possibilities for APs exists. APs could operate under direct control of the user, functioning similarly to an IdP, or they could be aggregators and providers of publicly-available information, or they could be business or government entities exchanging information under contractual agreements. Other viable models for APs will undoubtedly arise, as well. There are multiple modes in which attribute information can be gathered from APs and supplied to the SP. In one mode, the user can retrieve the (digitally signed) attributes and present them to the SP. In a second mode, a SP can be responsible for collecting attributes. This can be done under explicit authorization from the user via a cryptographic token given by a user to the SP and forwarded to an AP. Alternatively, the SP might retrieve attributes from APs that hold publicly-available information about the user, or it might contact APs with which it has contractual agreements, and thus not require explicit authorization from the user. A final mode of operation uses a combination of user-supplied and SP-retrieved attributes. For example, the user might present static attributes in the form of a digital credential, and the SP might be responsible for querying dynamic attributes. An AP can belong to a broad range of entities. For example, an AP could be a medical licensing board that can certify the role of medical professionals like doctors, EMTs, and nurses, or a location service that can certify the current location of the user, or an employer certifying its employees' association with the organization. In the case of the licensing board, the attribute has long term validity. It can be pre-fetched and stored by the

user. However, highly dynamic attributes like location must be fetched in real time.

Another entity in the authorization ecosystem that provides credentials is the user's identity provider (IdP). The IdP verifies a user's identity and asserts it to different authorities like the SP on demand. The trust is brokered using the PKI infrastructure through a certificate authority (CA), which is trusted by all the entities like the authorization system at the SP, the user and the IdP. The usual practice for large IdPs is that they themselves act as a CA and their public key is known and trusted by all the relevant entities.

As we already discussed, the SP typically provides some service or shares some sensitive data. In Figure 1, we show some of the service options that a SP can provide. It can be an online book store or a data repository or a file system, whose resources are protected by the authorization system. A SP can actually provide any conceivable service and these are just a few example of some of the common services.

Another important component in this system is the authentication service. Authentication can be done in several ways using username-password, public-private keys, identity certificates or even using certified attributes. In reality, username-password remains the most common authentication system in practice followed by identity certificates. Although this mechanism is not explicitly shown in Figure 1, it is assumed that the user does some kind of authentication to identify himself in the system.

Finally, we have data or resource providers in the system. They are the source of data or resources to the service provider. For example, if the SP stores and shares sensitive medical information about the patients, then the healthcare providers, pharmacies, personal health record repositories, and doctor's office all become data providers. It is important for the SP to either have a pre-established trust relationship with them or to have some method to cryptographically verify that the data indeed came from the claimed source.

1.3 Current research

The current research is focused on two areas in attribute-based authorization systems. First, specifying authorization policies in multi-authority systems where there are multiple stakeholders in the disclosure of sensitive data. The research proposes to consider all the relevant policies related to authorization in real time upon the receipt of an access request and to resolve any differences that these individual policies may have in authorization. Second, to enable a lot of entities to participate in the authorization process by asserting attributes on behalf of the principal accessing resources. Since it is required that these asserted attributes be trusted by the authorization system, it is necessary that these entities are themselves trusted by the authorization system. We propose a framework for service providers to evaluate trust in these attribute providing entities in real time. These two focus areas are elaborated in the following paragraphs.

The primary contribution of the proposed research focuses on issues in multi-authority systems. We address issues like considering relevant policies from all relevant authorities to ensure proper conformance before sensitive data can be released from the system. Since these policies which are specified by multiple authorities to protect the same resources in the system, they can have conflicts in their rules. Detection and resolution of these conflicts is required in these systems. We address these issues in our proposed framework by extending the set of standard combination algorithms with some new algorithms. Another issue we address in these authorization systems is that even though the access environment for many sensitive electronics resources like medical records or financial records is dynamic, the authorization systems protecting them are static, meaning that the policies and their combination do not change according to the access environments. Although some research models do propose some rules based on dynamic attributes, there are no systems which can completely adapt to the access environment. Our proposed framework enables

specification of a number of policies from each authority and use different policies for different environmental situations making the system completely adaptable to the access environment. Also, it proposes a mechanism to change the algorithm used for policies conflict resolution based on the access environment. For example, the authorization system can have a stringent authorization mechanism for normal situation, a lenient mechanism for emergency situation and a very strong authorization mechanism in the wake of attacks on the system. In the third and extreme case, the system can even make the protected resource unavailable temporarily until some corrective measures are taken. The access environment can be defined in a variety of ways using different attributes and the authorization system can adapt to that. This framework also proposes a mechanism to modularize the policy system by combining policies based on certain common attributes and evaluating them only when these attributes are true. Using this mechanism, we not only increase the operational efficiency of the system but also modularize it for easy analysis and maintainability. This also makes it possible for any authority to add and remove specialized policies using some spatio-temporal attributes to include them into the set of applicable policies to be evaluated when desired without compromising the general efficiency of the system.

To highlight the novelties of this framework it has been implemented in several prototype and open source systems which is presented in Chapter 6.

Another important problem limiting the functionality of attribute-based authorization and their acceptance in a large variety of service providers is that current authorization systems receive attributes only from entities that have a pre-determined trust relationship with it. Some examples of attribute providers are shown in Figure 1. In most practical systems the authorization system accepts attributes from the organization's LDAP server or a few identity providers (IdP) which provide a few identity related attributes for a principal. To enable a rich attribute-based authorization system, it is desirable that a large number of user attributes are available, possibly

provided by multiple entities. If a large number of attribute providers providing a wide variety of attributes are available, it would be very hard for a service provider to have a pre-established trust relationship with all of them. To leverage the benefits of these available attribute providers (AP), the service provider will be required to calculate trust in unknown entities in real time during a transaction processing and would need to accept attributes from entities which does not have a pre-established trust relationship with it. If the service provider does not have a good mechanism to evaluate the trustworthiness of these unknown APs, then it could be very easy for someone to set up a bogus attribute provider and have its attributes accepted. This threat could be addressed by using a community-based approach where a SP would accept attributes only from APs which are considered as trustworthy by members of the community. To evaluate trustworthiness of an AP based on the community's belief, we use a reputation system model.

We envision a future where attribute providers will be commonplace and service providers will face the problem of choosing one among multiple attribute providers that can provide the same user attribute. We address this problem by means of a reputation system model based on transitive trust. Entities express confidence in other entities to supply trusted attributes, forming chains from a service provider to different attribute providers. A service provider uses this transitive reputation to decide whether to accept a particular attribute from a specific attribute provider. We study different types of common attacks on reputation systems and the reputation system model in the proposed framework has been devised in such a way that it is resistant to these known attacks. The main threat that is addressed in this model is that malicious entities should not be able to subvert the system by either launching a slandering attack on a genuine attribute provider thus decreasing its reputation or launching a self-promotion attack to artificially increase the reputation of a malicious attribute provider. Other attacks like identity theft, malware compromising user

devices, or attacks on network infrastructure of the system are outside the scope of this model and we assume standard technologies to address these attacks. Also, it supports multiple attack resistant strategies to make the system adaptive to specific environments. Another problem that we address is that the user may be required to aggregate his attributes and present them to a service provider to prove he has the right to access some service. In Chapter 4, we propose a policy-based privacy enhanced framework for aggregating user attributes and evaluating confidence in these attributes.

1.4 Dynamic authorization framework

In current authorization systems, the attribute-based policies are predefined and are therefore static. In systems where sensitive information like financial or medical data is shared, the access environment is dynamic. Some systems try to account for this dynamic behavior by including special conditions for unexpected or dynamic environment but a large majority of systems don't have any mechanisms to do even that. The problem with such mechanisms is that they make the authorization policies very difficult to understand; they neither have fallback mechanisms in case of emergencies or exceptions, nor do they have provisions for using these conditions to define alternate policies for different access environments. To address these issues, we introduce our dynamic authorization framework, which has provisions to specify a number of alternate policies, all of which co-exist in the same repository. Which policy will be selected in real time can be defined in a meta-policy based on the dynamic environmental and user attributes. The framework also has an efficient mechanism to determine the set of relevant policies for an access request. This framework offers a number of advantages like -

- Different policies can be defined for different access contexts,
- If there are multiple authorization policies, the combination algorithm can

also be selected dynamically based on the dynamic user and environmental attributes,

- The authorization system will be modular and analyzable,
- It is very easy to include temporary or special purpose policies without the need to change the overall policies. This reduces the risk of mistakes or the chance of forgetting to revert back to the old policy when the temporary policy expires,
- And finally the mechanism to determine the set of relevant policies for an access request is very efficient and is 4-9x faster than the current mechanisms in determining the set of relevant policies to make an authorization decision for an access request.

1.5 AttributeTrust framework

In this section, we briefly introduce the AttributeTrust framework. We discuss this framework in detail in Chapter 4. We discussed in Section 1.2 that in order to evaluate the attribute-based policies, we need trusted attributes to use them as parameters in policy evaluation. In systems where there is a pre-established trust relationship between the attribute provider and the authorization system, the attributes can be verified easily. In practice, there are very few attribute providers and enterprise systems typically depend on the LDAP server to provide trusted attributes. While this approach is workable in closed or federated systems, it will not work in open systems. Even in closed or federated systems, relying on the LDAP server limits the number of authorization related attributes which can be used in policies. To really derive the benefits or flexibility and expressiveness of attribute-based systems, it is important that a lot of attributes be available for authorization. The problem in this case arises when the APs do not have a pre-established trust relationship with the SP (and his authorization system). To address this problem we introduce

AttributeTrust, which is a framework for aggregating user attributes, performing policy based trust negotiations, and evaluating trust in attributes. AttributeTrust presents some methods to aggregate attributes from a number of APs, if all the required attributes for a single authorization request are not provided by a single AP. The framework also uses a reputation system model for evaluating the transitive trust that an authorization system has in the AP. We studied the various types of attack against such systems and the AttributeTrust framework is carefully designed in such a way that it is attack resistant against the commonly known attacks.

1.6 Research motivation

Attribute-based authorization systems claim to offer numerous benefits over current systems but these benefits are not realized in current implementations because some underlying research issues limit the current implementations. These underlying issues are related to a scarcity of verifiable authorization related attributes, accommodating dynamic attributes, and leveraging dynamic attributes to provide context aware authorization. The current research investigates some of these issues as described in the following points -

1. In attribute-based systems, the authorization system should have access to trusted attributes to evaluate the policies. Access to these trusted attributes is a major challenge because there are very few entities which are pre-trusted by the authorization system and evaluating trust in attributes provided by unknown entities is an open question.

2. Using dynamic attributes for authorization and leveraging them to provide context aware authorization is a challenge in these systems. This is primarily because the authorization system needs to have different policies for different contexts and it needs to dynamically switch the authorization context considering the relevant attributes.

3. Modern authorization systems that provide access to sensitive medical data typically have a number of stake holders in maintenance, storage and disclosure of this data, each of which defines an authorization policy. One of the challenges in this context is to identify and combine all the relevant policies in runtime and to resolve the conflicts in the decisions arising from each of these individual policies.

1.7 Contributions

The main contributions of this research are summarized below -

1. A novel attribute-based framework for selecting relevant policies efficiently considering the dynamic context of the access request. This framework provides an efficient indexing method to select the policies applicable to each access request providing up to 9x advantage in speed of access and evaluation compared to current systems. This framework and its implementation are discussed in Chapter 5.

2. A novel method to dynamically change the algorithms to combine attribute-based authorization policies from multiple sources based on the context of the access request. This functionality enables the policy composing authorities to compose a number of co-existent dynamic policies and which specific policy is applied during each access request depends on the dynamic context of access. This method and its implementation are discussed in Chapter 5.

3. Several proofs of concept for the proposed attribute-based authorization system are developed in several different contexts, like electronic medical records, medical databases containing sensitive information, and rich-presence information. This authorization system provides a number of functionalities like fine-grained authorization, data hierarchy based protection, context aware authorization and

multi-authority authorization. The systems developed for providing attribute-based authorization for these applications is discussed in Chapter 6. Another proof of concept was developed by integrating the dynamic authorization system with the open source implementation of the Nationwide Health Information Network called CONNECT. This integration opens doors for offering this framework or any other services developed on this framework to be used by the public. Details of this integration are given in Section 6.3.

4. A novel attack resistant reputation system based framework for evaluating non-binary trust in aggregated subject and environmental attributes asserted by trusted or untrusted entities. These trusted attributes are necessary for an authorization system to evaluate attribute-based policies and the proposed framework provides this functionality. The current attribute-based systems only use attributes from pre-trusted entities and mostly from a single source. This framework greatly enhances the functionality and capabilities for an attribute-based authorization system. The framework is presented in Chapter 4.

The remainder of the dissertation is organized as follows -

- Chapter 2 presents some background in the area of authorization systems,
- Chapter 3 presents the related work in the areas of authorization systems, attribute-based authorization, and policy-based authorization systems for sensitive medical data,
- Chapter 4 presents the framework for evaluating trust in aggregated attributes,
- Chapter 5 presents the framework for dynamic context-aware authorization and dynamic policy conflict resolution,
- Chapter 6 presents the implementation of the proposed attribute-based authorization frameworks in various applications,

- Chapter 7 concludes the thesis and discusses some future work in this area,

CHAPTER II

BACKGROUND

2.1 Review of authorization technologies

Authorization systems are generally categorized either based on the method used for implementing access control or based on the entity which enforces access control. Using the former technique, they can be broadly categorized as Identity-based, role-based and attribute-based access control systems. Using the latter technique, they are categorized as discretionary access control (DAC) and mandatory access control (MAC) systems. In the following sub-sections, we will review these authorization technologies, except attribute-based access control, which we cover in detail in Section 3. In the following sub-sections, we will focus our attention on the advantages and limitations of these systems.

2.1.1 Identity-based authorization

In identity-based authorization systems, privileges to access resources in the system are given to identities of principals. A typical rule in an identity-based system would look like 'Principal p can access resource r with access rights a'. Identity-based systems can be further classified into two types - capability-based systems and access control lists.

1. *Capability based systems* - In capability-based systems, each principal holds a list of his access control rights over a set of resources. These rights are given to him in the form of capability certificates. When the principal wants to access a resource r, he presents the corresponding capability certificate to prove his access right. In this system, we get a user centric view of access control and

hence it is easy to analyze what resources a principal can access. On the other hand, it may be really hard to analyze which principals can access a resource as that would require checking the capability certificates of each principal.

2. *Access control lists (ACL)* - Access control lists are created for each resource or resource group. ACLs can be created with principals or groups of principals. In the former case, each ACL contains the list of principals who can access the resource or resource group that is protected by the ACL. A principal proves his identity during authentication and can access any resource if he is listed in the ACL protecting that resource. In the latter case, a principal is a member of a group and his identity to group relation is established upon authentication. In this case, the ACL contains a list of groups which can access the resources. In this case the model is not strictly identity based, since groups are commonly associated with role-based systems. In most practical systems, a hybrid model is used where the list contains individuals and groups. ACLs pose the reverse problem compared to capability certificates, i.e. it is easy to protect resources by listing the principals but it is hard to figure out the list of resources a particular principal can access.

2.1.2 Discretionary access control (DAC)

In Discretionary access control (DAC) the owner of the object specifies the access policy, listing who is allowed to access the resources and their corresponding access rights. In DAC the creator of the object is the owner by default and he can delegate his ownership rights to another principal [16]. The DAC model can be implemented using ACL or capability certificates. In the capability-based model, the capability certificates are created by the resource owner. Although this system provides great flexibility in defining access control policies, it also makes it hard to verify the security policies of the overall system. This is primarily due to the fact that resource owners

control and specify security policies. Another problem is that DAC is more prone to errors or misconfigurations in security policies and hence more susceptible to exploits.

2.1.3 Mandatory access control (MAC)

In Mandatory access control (MAC) the access control policies are defined by the system administrator. It is implemented as a multi-level access control system often containing highly sensitive data. It has several hierarchical classification levels and each resource and principal in the system is classified as a member of one of those levels. The principal's classification specifies his access level whereas the resource's classification specifies the minimum level of access a principal would require to access that resource. Examples of MAC are the Bell-LaPadula confidentiality model [29] and the Biba integrity model [34]. MAC requires that certain functional components like the operating system and associated utilities be 'trusted' and placed outside the MAC model because they are required to access resources at each access level. This makes it impossible to model a complete system using MAC without assuming that certain components are completely trusted.

In computer security, the principle of least privilege requires that a principal should be able to access only resources which are required for its legitimate purpose. Since the MAC model is based on a few distinct levels, it does not provide fine-grained control to satisfy this requirement completely. Separation of duty (SoD) is another principle which requires that the same principals are not given the privilege to execute transactions which are mutually exclusive from the security point of view, especially in the context of avoiding fraud. SoD can either be static or dynamic. Static SoD can easily be achieved by assigning principals privileges from only one group of mutually exclusive transactions. In practice, such a system is very inefficient and a more common approach, called dynamic SoD, is to assign principals privileges from multiple groups but restrict them to execute transactions from only one group during

system execution. Since MAC assigns fixed security levels to principals, dynamic SoD cannot be achieved in MAC.

2.1.4 Role based access control (RBAC)

Role-based access control was developed as an authorization system to be used by organizations, where employees were given access rights to the organization's resources. Employees in an organization who perform similar duties, need similar access rights to same resources. It was proposed that they should logically be part of a group and access rights should be given to groups. To simplify access management, employees were designated as members of some predefined 'roles' and access permissions were granted to these 'roles'. Using this system access management becomes easier as it is just a matter of adding or removing an employee to a 'role'. This also alleviates the problem of modifying a large number of access control rules when an employee joins or leaves the organization. The challenge with RBAC systems is that role management is a huge task in a large system and is mostly implemented in a centralized manner. Even if de-centralized administration is used, role management is an admin function and hence relies on an admin to manage and administer roles. Another problem with RBAC systems is that for each new composition of users, a new role must be defined. In a large system with large combinations of principals, the RBAC model results in a problem called 'role explosion' where the number of roles increases exponentially [42] and ultimately becomes unmanageable.

2.2 Authorization attributes

Subjects are principals who try to access resources to perform their tasks. Attributes are characteristics associated with the subject, resource or environment of the corresponding system. Accordingly, these attributes are classified as subject attributes, resource attributes and environmental attributes. Subject attributes are defined as a collection of attributes associated with the subject that can be used to classify

certain authorization related properties of the subject. For example, the age of a subject may be related to the decision whether he can perform certain tasks in the system, like purchasing alcoholic beverages online, so a subject's age is likely to be an authorization related attribute. Such attributes are identified and listed in the system. Resource attributes are characteristics of resources that help in specifying authorization rules on the resource. For example, files can be categorized according to their sensitivity or their content. These categories are the resource attributes for this example. Referring to the MAC model, the sensitivity levels of subjects and resources are their authorization related attributes. In many cases, certain parameters of the external environment also have an effect on how an authorization decision is reached. These parameters form the environmental attributes for the system. For example, a subject may have access to resources during normal business hours of 9 am to 6 pm in his time zone but not outside of that time. In this case, time becomes an environmental attribute required to make authorization decisions. The authorization system usually has reliable communication channels with trusted entities that provide these environmental attributes.

2.3 Attribute-based authorization

Attribute-based authorization is a relatively new paradigm in authorization systems. In attribute-based systems all the authorization elements, i.e. the subject, resources, actions and environment are defined in terms of attributes. Attribute-based authorization policies contain resources specified in terms of their attributes, which can be accessed by a subject specified by his relevant attributes with actions specified by the action attributes under environmental conditions specified by the environmental attributes. Among these attributes, the resource and action attributes are internal to the system in the sense that these attributes can be defined within the system. There is no need for any entity to verify these attributes. On the other hand, subject and

environmental attributes are external to the system, meaning that they are provided to the system by external entities. As such the authorization system requires that these attributes and their values be asserted by trusted entities. In this research, we propose several methods for trusted entities to provide these attributes. This is discussed in more detail in Chapter 4.

2.4 Authorization policy languages

Modern enterprise IT systems have become very large and complex, which has also increased the complexity of its security mechanisms. Specifically, modern authorization systems are tasked to provide a large number of users secure authorization to a large number of resources which are often stored in distributed repositories. Also the requirements for authorization are large, so the resultant system becomes very complex. If this authorization system is developed as part of the application code, then it will be hard to analyze the system and to verify if all the requirements are being met. Also, each system will require its own customized authorization system developed for a specific application. To avoid these complexities, specific languages have been developed to specify authorization requirements. These languages have a special syntax for specifying the authorization requirements in an abstract fashion and their specific implementation can be customized to requirements of a single application. Some of the common languages, like EPAL [43] and XACML (Extensible Access Control Markup Language) [5], are XML based, but other proposed languages like SecPAL are defined more in terms of formalism with a suggested XML schema [82]. These XML based schemas make them especially suitable for use in web services. In fact, some new architectures are proposing to use these policy-based authorization systems to be deployed as a web service in a service oriented architecture (SOA) [17]. XACML is an OASIS standard for specifying access control policies [5]. It is an attribute-based authorization language where access control rules are specified as a combination of

subject, resource, action and environmental attributes. XACML is fast becoming one of the most popular standards in the industry as it supports flexible and fine-grained authorization policies. These rules are easy to compose and can be comprehended by human beings, since they are XML based. The current recommendation is XACML v 2.0 and XACML v 3.0 is a draft.

CHAPTER III

RELATED WORK

This research is in the area of attribute-based authorization systems. The main focus is on attribute-based policies used to protect sensitive medical data. In this chapter, we present related work from several background areas like authorization systems, reputation systems, and policy-based authorization systems for protecting sensitive medical data.

3.1 Authorization Systems

Authorization systems are an essential part of modern security systems. They mediate a principal's access to resources in the system. One of the first formal presentations of authorization mechanisms used in operating systems appeared in a paper by Butler Lampson [56], where he talked about different methods to protect digital resources in modern computer systems.

3.1.1 Butler Lampson's Protection

Butler Lampson's work on resource protection focuses on access control in the operating system level where he presents the properties of most known access control mechanisms at that time [56]. The main focus is to enforce the rules of modular programming to guarantee that errors in one module will not affect other modules and to enable use of proprietary software with execute-only permissions without errors that affect other programs. It uses a message passing model to illustrate the concepts and implementation of the model. It also proposed two important concepts called 'protection domains' and 'access matrices', which are used to enforce access control in computing systems. The idea of protection domains is that different protection

environments or contexts, have different levels of access control. For example, administrators and users in an operating system have different levels of access to resources. An access matrix is defined as a matrix where each column defines the access rights on a resource and each row describes the access rights a subject has on each of the resources. The element (i, j) in the matrix defines the access rights subject i has on resource j. The resources can be processes, domains, files, memory locations or other subjects. Although these concepts were proposed in the context of avoiding errors, they work equally well in case of malicious entities that try to either degrade the service or improperly gain access to resources in modern systems.

3.1.2 Nexus authorization logic

Nexus Authorization Logic (NAL) was proposed by Schneider et al. to provide a basis for specifying and reasoning about credentials and authorization policies [78]. It extends previously proposed access control logics based on "says" and "speaks-for" operators. Within this framework, the request authorization depends on (i) the source of the requester, (ii) the outcome of performing an analysis on the requester, or (iii) the use of trusted software to encapsulate or modify the requester. NAL extends Abadi's CDD [18] [19] access control logic by adding support for axiomatic, analytic, and synthetic bases for trust. It also adds two kinds of principals called groups and sub-principals in the authorization logic that helps bridge the gap between the abstractions found in Abadi's model and the details of actual software implementations. In order to evaluate the framework, they implemented an operating system called Nexus that supports encoding credentials and enforcing security policies specified using NAL [81]. The Nexus operating system provides operating system support for tamper-proof co-processors for secure computing. It uses a single hardware protected cryptographic key as its root-of-trust. To demonstrate NAL, they implemented a suite of document viewer applications that run on Nexus. One

application called TruDocs (Trustworthy Documents) controls the display of documents that contain excerpts whose use is subject to restrictions. Another one called ConfDocs (Confidential Documents) protects the confidentiality of documents built from text elements having security labels.

3.1.3 Role-based access control

Role-based access control (RBAC) is one of the most popular access control systems today. The concept of RBAC began with multi-user and multi-application on-line systems pioneered in the 1970s. It gained its prominence when it was adopted to model the enterprise workflow for assigning permissions to employees in an organization. Pioneering work in this area was done by Sandhu et al., [76] and Ferraiolo et al., [44] who developed practical models to be implemented in enterprise systems. These models were later consolidated as the NIST RBAC model [77]. Some more recent RBAC models are presented in [30], [24].

The basic concept of RBAC is that users are assigned to roles and permissions are assigned to roles. Users acquire permissions by being members of those roles. The user-role and permission-role assignment can be many-to-many. Three flavors of RBAC are used in most practical systems - Basic RBAC, Hierarchical RBAC and Constrained RBAC. Hierarchical and Constrained versions are based on the basic RBAC.

Basic RBAC - This flavor captures the essential elements to support RBAC. User-to-roles mapping is called user assignment (UA) and permissions-to-roles mapping is called permissions assignment(PA). This model is formally defined below -

- U, R, P, and S (users, roles, permissions and sessions respectively),
- $PA \subseteq P \times R$, a many-to-many permission to role assignment relation,
- $UA \subseteq U \times R$, a many-to-many user to role assignment relation,

- user : $S \rightarrow U$, a function mapping each session s_i to the single user $user(s_i)$(constant for the session's lifetime), and

- $roles : S \rightarrow 2^R$, a function mapping each session s_i to a set of roles $roles(s_i) \subseteq \{r \mid (user(s_i), r) \in UA\}$ (which can change with time) and session s_i has the permissions $U_{r \in roles(s_i)}\{p \mid (p, r) \in PA\}$.

Hierarchical RBAC - Hierarchical RBAC is based on the concept of role hierarchies as a natural means to reflect organizational hierarchies. The senior role has more authority than a junior role and inherits all the permissions of its juniors in the hierarchy. The formal presentation of Hierarchical RBAC is presented below -

- U, R, P, S, PA, UA, and $user$ are unchanged from the basic RBAC model,

- $RH \subseteq R \times R$ is a partial order on R called the role hierarchy or role dominance relation, also written as $\geq$, and

- $roles : S \rightarrow 2^R$ is modified from the basic RBAC to require $roles(s_i) \subseteq \{r \mid (\exists r' \geq r)[(user(s_i), r') \in UA]\}$ (which can change with time) and session s_i has the permissions $U_{r \in roles(s_i)}\{p \mid (\exists r" \leq r)[(p, r") \in PA]\}$.

Constrained RBAC - The Constrained RBAC model is used to model the natural constraints in organizational roles. For example, in an organization a purchase manager who makes purchase decisions is different from a finance manager who makes the payment. Another disjoint role is a finance auditor who audits the accounts. No single individual should be able to activate two or more of these roles, which is an essential constraint on these roles. These roles are also referred to as mutually exclusive roles. Another type of role in this model is the prerequisite role where a user has to be a member of a prerequisite role before he can become a member of the desired role. For example, in an IT company before a user can be mapped to a 'developer' or 'test engineer' role in a project, he should be a member of the role 'project member' for that project.

3.1.4 Purpose-based access control

Recently, researchers have shown interest in implementation of privacy policies which are typically expressed in terms of the purpose for which any sensitive data is used. This model, called purpose-based access control, is different from current access control models. Current models focus access control with the problem for providing access to authorized subjects or roles as opposed to the purpose of access in purpose-based access control. Byun et al. proposed a model in which the purpose information associated with a given resource specifies its intended use [37]. It supports multiple purposes to be associated with each resource. It also supports explicit prohibitions, which helps privacy officers in specifying that particular resources should not be used for certain purposes in any case. They define the 'intended purpose' (IP) of the resource, which is defined by the resource owner or administrator, and 'access purpose', which is specified by the requester. The model essentially matches these two to decide whether access should be provided. As discussed, the 'intended purpose' is of two types, 'allowed intended purpose' (AIP) or 'prohibited intended purpose' (PIP). In case of a conflict between the two, a 'denial-takes-precedence' policy assures that PIP takes precedence. Intended purpose is mostly defined ahead of time. In cases where organizations choose to apply additional policies like country specific policies, Children's Online Privacy Protection Act (COPPA) or individual customer's privacy policy, IP can also be determined during execution. In their model, users are explicitly required to state their 'access purpose', and they are trusted based on their roles, system and role attributes. The authors proposed a modified syntax for SQL queries to include 'purpose of access'. This query is intercepted at the access control level and the purpose is evaluated. If the query is legitimate, it is re-written in the standard SQL syntax and executed on the database.

3.1.5 Attribute-based encryption

Attribute-based encryption (ABE) was first proposed by Sahai and Waters in [75]. Although ABE is not an authorization system per se, it provides a method to cryptographically enforce access control policies. We discuss this system here because it is a cryptographic attribute-based access control system, where only principals which hold certain verifiable attributes with specific values can view secure messages. This system provides identity-based encryption where the identities are composed of a group of attributes. The basic idea of this system is to enforce a simple attribute-based policy cryptographically by encrypting the message using the relevant attributes. For example, in a computer science department, if the chairperson wishes to send a message to all the faculty members who are in the 'systems' specialization and on the faculty hiring committee, he can encrypt the message using identity attributes {Faculty_hiring_committee, Faculty, Systems, Computer_science}. Only individuals who possess all of these identity attributes can decrypt the message. The system also prevents the message from collusion attacks so a faculty on the hiring committee of the history department with identity attributes {Faculty_hiring_committee, Faculty, History} cannot combine his attributes with a student in computer science with identity attributes {systems, computer_science} to decrypt the message. One of the main advantages of this system is that it does not require the message to be stored on trusted servers and to be communicated on secure channels which give access only to authenticated individuals. The message can be encrypted using the desired set of attributes and communicated via unsecured channels and stored on untrusted servers, but it can only be decrypted by individuals with the desired identity attributes.

3.1.6 Attribute aggregation

Several researchers have looked into the problem of using authorization related attributes asserted by more than one identity provider [38] [54]. The main idea is

to implement some mechanism to aggregate attributes from a number of identity providers and then provide them to the authorization system to reach authorization decisions. Chadwick's model [38] deals more with the problem of linking different attribute authorities for a user while protecting his privacy. He advocates the use of existing technologies for implementing this model. Klingenstein [54] addresses this problem in the context of a federation and assumes that there is some mechanism in place to associate different identities that a user has registered with different attribute authorities.

3.2 Reputation systems

3.2.1 Transitive trust systems

In the transitive trust (TT) model, trust is brokered between two untrusted parties by a third party which is trusted by both the parties. Simply put, if A trusts B and B trusts C, then A trusts C. These trust chains can be arbitrarily long, the maximum length being defined by the source party A. The actual amount of trust that A places in C is calculated using a trust metric. In these metrics each user decides the amount of trust it places in its peers which is usually a non-binary value between 0 and 1. Using these values, a trust chain can be converted into a numeric trust value. These metrics have their limitations though. While it is easier to propagate trust, propagation of distrust is trickier. For example, if A distrusts B and B distrusts C, should A trust or distrust C? Another limitation is that the translation of trust values from one metric to another is generally not possible. Initially these metrics were proposed to support PKI based authentication by calculating the likelihood that a name to public key binding of a target party is true as claimed by the target party [72] [57]. More recent metrics are focused on very divergent areas: from establishing trust in e-commerce transactions to online markets to recommendation systems.

3.2.2 Ebay and Advogato

There are many online systems which use reputation systems models called recommendation systems to capture feedback from the user community to decide the trustworthiness of a user [3]. One of the most popular systems is used by the online trading site Ebay, where sellers and buyers provide feedback about each other. Each seller or buyer has an opportunity to provide a feedback score once they complete a transaction based on their experience and an aggregate score is visible next to their online profile. The detailed score report along with comments and value of transaction is available to users. Ebay's initial feedback system was highly criticized for being very simple and susceptible to manipulations. Ebay has improved its system since then with more stringent identity checks for registration, the addition of transaction values and other feedback details, and analysis of behavior anomalies to detect manipulations. Although it is still not perfect, Ebay's feedback system is very helpful for buyers and sellers and 71% of the users agree that it increases their confidence in the other party, which is crucial for online markets [2].

Advogato is a research testbed for trust metrics and social networking technologies [1]. This trust metric uses four key members as 'completely trusted' seed accounts and other users are trusted relative to their 'closeness' to these seed accounts. The core idea of the Advogato metrics was proposed by Levien et al. in [57]. The basic trust metric evaluates a set of peer certificates to accept a set of user accounts. These certificates are represented as a graph, with each account as a node, and each certificate as a directed edge. The goal of the trust metric is to accept as many valid accounts as possible, while also reducing the impact of attackers.

3.2.3 EigenTrust

The EigenTrust reputation management system is an algorithm to reduce the number of downloads of inauthentic files in peer-to-peer networks [53]. The open nature of peer-to-peer networks opens many avenues for attack and to introduce inauthentic files to degrade the quality of the peer-to-peer network. The EigenTrust model is based on power iterations of trust values and provides a secure and distributed algorithm to compute global trust values of users. In simulations, EigenTrust has been shown to significantly decrease the number of inauthentic files on the network, even under a variety of hostile conditions where malicious peers cooperate in an attempt to deliberately subvert the system. EigenTrust uses a matrix of normalized local trust values denoted as $c_{i,j}$, where each member of the network is an element in the matrix. A member asks his friends at one hop distance about their perception of his friend's $c_{i,j}$ values and formulates his matrix. He then asks his friend's friends at two hop distance from him about their opinion of each member and then evaluates his trust matrix using power iterations. If we consider a sufficiently large number of hops, then each member's trust matrix converges to the same value, where the left principal eigen vector represents the global trust values for each member in the community.

3.2.4 Propagation of trust and distrust

Propagation of trust and distrust was studied by Guha et al., in [46]. They developed a framework of trust propagation schemes and evaluated the schemes on a large trust network consisting of 800K trust scores expressed among 130K people. This model can predict the trust values between any two members of the network with very high accuracy based on just a small number of trust or distrust values per member. One of the differentiating factors of this model is that it includes the notion of distrust. This model has trust and distrust matrices, which are developed by capturing atomic propagation of trust and distrust. These matrices are consolidated

into a belief matrix, which captures the user's belief about the trustworthiness of other users. The authors used the test data available from Epinions.com to test their model and discovered that based on a small number of available trust and distrust ratings, their model can predict the unavailable trust or distrust between any two users in the network with high accuracy.

3.3 Policy-based systems for protecting sensitive data

3.3.1 Flexible support for multiple access control policies

Although several access control policies can be devised for a system, in reality only a single access control policy called the closed policy is used in systems. Jajodia et al. address this problem by proposing a framework that can enforce multiple access control policies within a single framework [49]. Their framework is based on a specialized language, which the policy enforcement entities can use to specify multiple policies for a single system. The language provides specification of positive and negative authorization rules and provides support for conflict resolution and decision strategies. The main advantage of this system is that all the policies can co-exist in the system and can be applied by the same enforcement agent depending on the specification. In this framework, authorization rules can be defined to grant access to users, groups or roles. It also supports hierarchy among authorization subjects. Since the framework supports both positive and negative authorization, there is a chance that conflicting rules may be created in the system for the same access request. To address this, the framework supports conflict resolution strategies for resolving conflicts in real time. Specifically it supports four strategies - a) *No conflicts*, which does not allow conflicting rules to be created in the system, b) *Denials take precedence*, which means that the occurrence of a single deny rule will make the final result 'deny', c) *Permissions takes precedence*, which means that occurrence of a single permit rule will make the final result 'permit', and d) *Nothing takes precedence*, which means

that in case of a conflict the final decision is deferred to specialized policies called decision policies. These decisions policies can in turn be open or closed. The open policy is analogous to the 'default permit' system which permits all accesses unless there is a specific rule denying the access. On the other hand, the closed policy is analogous to the 'default deny' system which denies all access requests unless there is a specific rule permitting the access. Most authorization systems which protect sensitive data implement a 'default deny' strategy. Other notable works in this area include [62], [63], [52].

3.3.2 Supporting multiple access control policies in database systems

Another multi-policy access control system for databases was proposed by Bertino et al. in [32]. This system provides the capability to protect different tables in a database with different policies. The system can be implemented as 'open system' or 'closed system'. This framework supports both positive and negative authorizations and as such, there is scope of defining conflicting authorization policies for the same resource. To address this issue, the framework also has support for conflict resolution. The central notion for conflict resolution in this system is that there are two types of authorization, 'strong authorizations' which do not permit creation of conflicting policies in the system, and 'weak authorizations' which permit creation of conflicting policies and resolve them at runtime. Authorization rules can be overridden and as a general rule, strong authorization rules override weak authorization rules. To resolve conflicts among two weak authorizations, the system default is defined based on whether the system is an 'open system' or a 'closed system'. A user may choose a different resolution strategy from the system default to resolve conflict between two weak authorizations.

3.3.3 Policy Engine Evaluation

In this section, we will focus our attention on some works focused on evaluating and improving the XACML engine's Policy Decision Point (PDP). The performance evaluation and improvement of PDP received little attention in the research community until now but as XACML is becoming a popular standard for implementing authorization systems, the performance of PDP will soon become a bottleneck for enterprise grade systems. Turkmen et al. compared the performance of PDPs of three open source XACML engines, namely Sun XACML, XACMLLight and Enterprise XACML [84]. The comparison is based on the load and evaluation stages of the PDP engine. In the loading stage, the XACML engines load the policies into the cache from the file system where they are stored. In the evaluation stage, the cached policies are accessed and evaluated against the access requests. During loading comparison, XACMLLight performed the best because it uses Java Beans framework to load and store the policies. This was followed by Sun XACML which loads all the policies. The worst performer was Enterprise XACML, because it uses advanced methods like target and policy indexing and result caching. Since these methods are time consuming, they contributed to Enterprise XACML's poor loading performance. On the other hand, during the evaluation stage Enterprise XACML's advanced methods paid off and its performance was the best. Sun XACML followed in performance and XACMLLight's performance was the worst. Even the memory requirements of XACMLLight are pretty heavy and a test bed with 2GB RAM with 1GB of heap space could not provide enough memory for it to load and evaluate 10,000 policies during the performance evaluation. In another work, Liu et al. proposed XEngine, which is an improved implementation of an XACML engine using a number of optimizations [25]. They compare XEngine's performance with Sun's open source XACML implementation and report the results. In XEngine, the XACML policies are first converted to a numerical form and then normalized. Then this normalized numerical policy is

converted into tree data structures for efficient processing. Using numerical policies, XEngine can use efficient integer comparison methods during evaluation compared to the inefficient string comparison methods. The limitation with this approach is that all the policies in the original XACML policies are converted to first applicable. This helps in more efficient processing because the engine is not required to process all parts of a policy or policy set but on the downside it does not provide the functionality of other combination algorithms, which is a major advantage of XACML. Performance comparison results show that XEngine's performance is better than the Sun XACML engine by three to four orders of magnitude for large number of policies.

3.3.4 Hippocratic databases

Hippocratic databases are privacy preserving databases which are built on the foundation that information should be shared only with consent for the purpose it was collected. Agrawal et al., proposed these databases and identified the challenges associated with implementing them in real database systems and some possible solutions [21]. Hippocratic databases are different from either statistical or secure databases. They borrow a lot of basic learning from both but have to cater to a much wider range of queries and have to confirm to stricter design and usage requirements. They have to conform to ten basic requirements - *Purpose Specification*, *Consent*, *Limited Collection*, *Limited Use*, *Limited Disclosure*, *Limited Retention*, *Accuracy*, *Safety*, *Openness* and *Compliance*. To provide these features, Hippocratic databases perform privacy checks in queries before, during and after execution. Before execution it checks if the query being executed conforms to the stated purpose and the querier is allowed to run that query. During execution the Record Access Control ensures that only records whose purpose attribute includes the query's purpose will be visible to the query. After query execution the Query Intrusion Detector detects

any anomaly in the query results and tries to flag queries whose results look anomalous. This work also identifies the main challenges in implementing this system. To name a few, identification of privacy policy languages to describe the privacy policies, efficiency of the system in the presence of additional privacy checks, limited use/collection/retention/disclosure of sensitive data, methods to provide safety while maintaining openness, and maintaining compliance with policies are discussed. Hippocratic databases achieve their goal of limited disclosure by a number of mechanisms. They can perform 'strict cell-level' access control enforcement, as compared to row/table/column level enforcement in other databases. To achieve this, a mask is used which sets all the inaccessible cells to null. They can also perform tuple-level, table semantics limited, query semantics level or application level access control enforcement. Some applications of Hippocratic databases are discussed in [21], [22], [28].

CHAPTER IV

ATTRIBUTETRUST

To enable a rich attribute-based authorization system, it is desirable that a large number of user attributes are available, possibly provided by multiple entities. The user may be required to aggregate his attributes and present them to a service provider to prove he has the right to access some service. In this chapter, we present AttributeTrust a policy-based privacy enhanced framework for aggregating user attributes and evaluating confidence in these attributes. We envision a future where attribute providers will be commonplace and service providers will face the problem of choosing one among multiple attribute providers that can provide the same user attribute. In AttributeTrust, we address this problem by means of a reputation system model based on transitive trust. Entities express confidence in other entities to supply trusted attributes, forming chains from a service provider to different attribute providers. A service provider uses this transitive reputation to decide whether to accept a particular attribute from a specific attribute provider. We discuss how the AttributeTrust model prevents common attacks on reputation systems. AttributeTrust differs from the current approaches by deriving its attack resistance from its specific context of attribute provisioning, its voting mechanism formulation, and unique properties of its confidence relationships. The main threat that is addressed in this model is that malicious entities should not be able to subvert the system by either launching a slandering attack on a genuine attribute provide thus decreasing its reputation or launching a self-promotion attack to artificially increase the reputation of a malicious attribute provider. Other attacks like identity theft, malware compromising user devices, or attacks on network infrastructure of the system are

outside the scope of this model and we assume standard technologies to address these attacks.

4.1 Introduction

Attribute-based systems provide a highly granular, scalable and semantically rich method for access control and authorization [91]. A service provider (SP) defines policies, which dictate the set of attributes required for accessing its services. The user provides credentials that contain verifiable attributes to gain access to these services. Different levels of access can be provided based on what attribute values the user possesses. This scheme is especially suitable for distributed systems, where maintaining real time synchronization among access control lists is a huge problem [68]. Another merit of this access control model is that, it does not require a priori knowledge about the user at the service provider. Although attribute-based systems have many advantages compared to traditional systems like role based access control [91], they have some practical limitations. We examine some of these limitations as a way to motivate AttributeTrust. Firstly, it is very difficult for a service provider to directly verify each users attributes in real time. Traditionally, the problem of verification is addressed by requiring users to get their attributes bundled into credentials, verified and digitally signed by an Identity Provider (IdP) [70] [61]. These credentials are trusted by the SP (which acts as a relying party). We argue that this approach limits the attribute set that can be used because IdPs might not be willing to certify certain attributes or it might not be efficient for them to verify these attributes. For example, the IdPs may be unwilling to certify attributes that are not identity related. Another problem is with rapidly changing attributes or adding new attributes. Fresh credentials need to be issued by the IdP every time an attribute value changes or a new attribute is added. Secondly, the abovementioned scheme also requires that all relying parties have reliable access to all the IdPs public keys [72]. This scheme can

either have only a few IdPs effectively overloading them and making them lucrative targets for attacks or have a high number of IdPs, making it difficult for each relying party (RP) to know every IdPs public key reliably [72] [33]. We argue that this represents a performance bottleneck in current systems. Thirdly, in current systems, only a few user attributes are available. In order to achieve a high granularity in attribute-based systems, a large number of attributes are desired. Since an attribute is any parameter that characterizes a user, a single IdP may not be able to verify every attribute and issue credentials. If there are multiple IdPs, the user may be required to aggregate these attributes before presenting them to the SP [38] [54]. Fourthly, most current systems use a group of attributes contained in a credential for supplying user attributes. AttributeTrust provides a higher granularity by dealing with individual attributes. This enhances user privacy by limiting the unnecessary disclosure of attribute values. Finally, we discuss the problem of how much trust a RP should place in attributes. This problem is separate from verifying the credentials themselves. IdPs have different certification policies, making the attribute verification strength vary from one IdP to another. If all credentials are trusted equally, malicious users would get their credentials issued by IdPs with lenient certification policies with little verification. They may also try to subvert the system by registering as an IdP themselves and issuing bogus credentials to other malicious users. This problem is illustrated by an example. Alices local social club, her university and the state department of driver services can each act as an IdP and issue credentials certifying Alices date of birth. The RP can verify all the three credentials using the respective IdPs public keys but the question is which IdP can the RP trust to provide this particular attribute?

AttributeTrust This chapter presents AttributeTrust, a framework for aggregating user attributes, performing policy based trust negotiations, and evaluating trust in attributes. To address the problem of verifying a large number of attributes

by the IdP, we introduce the concept of an Attribute Provider (AP), by which we mean something different from some prior usages of this term [65]. Our definition of an AP is an entity that holds user attributes and can provide them to the RP in a reliable way. Traditionally, the functionality of the IdP is limited to providing signed user credentials, whereas the AP can operate in multiple ways to provide attributes reliably. First, the AP may provide signed credentials to the user in which case its functionality is similar to an IdP. Second, the AP may operate a public database holding user attributes; the user first proves her identity to the SP and points to her attributes in the APs public database. Since the AP server is a trusted entity, attributes may be fetched from the server in a reliable way. Finally, the AP may store users attributes in a secure database. The user provides the SP with a cryptographic token to be presented to the AP to release the attributes. The AP provides enhanced functionality compared to a traditional IdP. We discuss the advantages of the APs enhanced functionality in Section 4.2. Figure 2 shows the basic components of the framework. Entities are the User, the Relying Party and multiple Attribute Providers. The user requests some service from the RP. The RP initiates a trust negotiation, where the user and RP both are required to provide attributes based on the other partys disclosure policies. To provide these attributes the user may aggregate some or all these attributes from the APs himself (referred to as self aggregation) or provide a token to the RP to fetch some attributes from the AP directly (referred to as delegated aggregation). The RP also provides its attributes to the user, which may be aggregated. The RP and the user accept the other partys attributes based on their trustworthiness (the method for evaluating the trustworthiness is explained in the next paragraph). By providing the attributes, the RP and the user satisfy the other partys disclosure policies, thus completing the negotiation successfully. When the trust negotiation is successful, the RP provides the service to the user. It is important to distinguish between trust in the paths leading to the attributes vs. trust

in attributes themselves. In AttributeTrust, each APs public key is stored such that they are readily accessible to all the entities, so the attributes can be verified unambiguously by the RP. On the other hand, the RPs trust in attributes provided by an AP is calculated by using a reputation system model. The system is represented as a graph where the nodes represent entities that may be users, RPs or APs and the directed edges represent the confidence the source node has in the destination nodes ability to provide trusted attributes. When the RP wants to calculate its trust in attributes provided by an AP, it calculates a confidence value based on all the confidence paths leading from itself to the AP. If the RP does not directly trust the AP, it may have a trusted peer who has trust in attributes provided by the AP. In this way, chain of trust helps in calculating the RPs transitive trust in the AP. Even if the RP has a direct trust relationship with the AP, it is better to calculate a final trust value based on all the available confidence paths, so that the RP uses his and his peers collective trust to calculate his final trust. This final trust value can be used by the RP to determine if it can accept the attribute from this AP. The primary advantage of this scheme is that any entity can serve as an AP in the system; the RP will accept attributes from it only if the AP is in the RPs 'circle of trust'[1]. The details of the model are presented in Section 4.3. We have studied some common attacks on reputation systems and made the model robust against such attacks. These attack resistant properties are discussed in Section 4.4.

The main contributions of this framework are defining the scope of an APs functionality, providing a framework for aggregating attributes from APs for use in trust negotiations and building an attack resistant reputation system for evaluating trust in aggregated attributes. The rest of the chapter is organized as follows Section 4.2 discusses attribute aggregation and its use in trust negotiations. Section 4.3 presents

[1]Circle of Trust means all the nodes covered in a circle with RP at the center and the radius is equal to the maximum path length (L)

details of AttributeTrust, while Section 4.4 studies the attack resistant properties of AttributeTrust. Section 4.5 discusses how AttributeTrust satisfies some of the model requirements noted in Section 4.3.1 and finally Section 4.6 discusses related work.

4.2 Attribute Aggregation and Trust Negotiation

In this section, we discuss the concept of Attribute Provider (AP), methods of aggregating attributes from APs, and how to use these attributes in automated trust negotiations.

4.2.1 Attribute Provider

In Section 4.1, we noted that the AP is an entity that holds user attributes and can provide them to the RP in a reliable way. We also presented a number of ways in which an AP can function. In this way, the AP encompasses some of the functionality of an IdP and also extends it in a way that allows any entity to set up an attribute server and register as an AP in the system. This is useful because now an AP provides attributes with which it is closely related (hence easily verifiable) and saves the IdP the trouble of verifying obscure (from the IdPs perspective) attributes. For example, whether Alice, who is a music major, is an expert or amateur piano player is something that can be asserted by her music school but it might be difficult for a traditional IdP to assess. This attribute may be of interest to a high school looking for a piano teacher. The main advantage of using this system is that a large variety of attributes can be provided and it can be applied in several new dimensions rather than just proving a users identity. Whether the music school is qualified to provide this attribute is decided by the reputation system discussed in Section 4.3.

4.2.2 Attribute Aggregation

Any user in this system will usually have a number of attribute providers. While building mutual trust with a RP, the user is expected to provide a number of attributes [54]. These attributes need to be aggregated from APs. Depending on the APs functionality, this may be done by the user or delegated to the RP by the user.

4.2.2.1 User mediated aggregation

This case applies when the AP issues signed credentials to the user. The user can have pre-issued credentials or can request freshly signed credentials. Minimum information disclosure can be applied where the AP can provide credentials with the attribute values hashed by a one-way function [27]. The user can provide the relevant attributes to the RP in plaintext, which can be verified by the RP by hashing them with the same one-way function and comparing against the signed credential [27] [83].

4.2.2.2 Relying Party mediated aggregation

The relying party can be delegated the task of aggregation by the user. This can be done in two ways. First, if the AP runs a public database, the user points the RP to a particular pseudonyms attributes in the APs database. The user can establish ownership of the pseudonym in multiple ways like providing an identity certificate, through previous relationship with the RP, or correlating other data the RP maintains about the user. Second, the AP may be running a database answering queries from authenticated users. In this case, the user can delegate the aggregation by providing a signed cryptographic token to the RP, which can be presented to the AP for authentication and requesting the AP to release particular attributes. One way to construct such a cryptographic token is discussed here. The user and AP set up a shared key. The user creates the token by specifying a particular set of attributes and encrypting them with the shared key. The advantage of using shared key is that this key can be changed independently for each AP, in case of any security breach

at the AP. In our example, the music school is the AP and Alice is the user. Alice creates a delegation token with key K_{ALICE} for the attribute PianoPlayer as shown Token = {E(Attribute=PianoPlayer, K_{ALICE}), ALICE }

4.2.3 Trust Negotiation

When the user contacts the relying party requesting a service, the RP initiates an automated trust negotiation based on policies. The RP and the user both have attribute disclosure policies, meaning the disclosure of sensitive attributes are protected by policies. In a particular scenario, suppose the RP asks the user for a sensitive attribute. The user may have a policy that requires that the RP first prove its identity before the attribute can be released. For example, when the high school requests Alice to prove that she is an expert piano player, Alice may have a policy in place that requires the high school to prove its accreditation by the competent authority. These disclosure policies are configurable and can be dynamically updated. The user and the RP each have a negotiation policy tree similar to the one shown in Figure 3. In a negotiation tree of depth n, an attribute at level k can only be disclosed when all the required attributes from level n up to level k+1 have been presented by the other negotiating party. The policy tree shown in Figure 3 is an AND-OR tree defining the release requirements for attributes. These requirements are expressed as AND ($\wedge$) and OR ($\vee$) relations among the nodes at the next level in the tree. The OR relations are shown by dashed lines whereas the AND relations are shown by solid lines. Both the trees at the user and the RP side grow dynamically when each party learns about the other party's release policies for that attribute. For example, for releasing AU1 the user may require that either $AR1 \vee AR2 \wedge AR3$ must be released by the RP. The RP in turn requires that the user release other attributes before it can release attribute AR1 or attribute AR3. A requirement at a node is satisfied when the policies of all its children are satisfied. The release policy of the root node

can be satisfied if a path from any leaf node to the root node can be found where the policies for all the intermediate nodes as fulfilled. Only then the negotiation is successful. If no such path can be found, the negotiation is unsuccessful. In case of circular dependencies in the policies, a proof of holding the attribute can be used to break the deadlock. For pre-signed credentials, this can be done by hashing out the attribute values in the credentials and presenting the credentials [83]. We extend this case for the AP. Pre-signed credentials can be used as described above. If the AP hosts a public database, the user can point to his entry in the database. In case of protected databases, the user can create a token requesting the AP to assert that the user holds the required attribute and present this token to the SP. SP can present this token to the AP to receive this assertion.

4.3 AttributeTrust

In this section, we present the AttributeTrust framework for measuring confidence in aggregated attributes. We use a reputation system model, where each member in the system votes for his peers ability to provide trusted attributes. In Section 4.3.1, we discuss some desirable properties of the reputation system model. In Section 4.3.2, we show the basic formulation of the proposed model. Section 4.3.3 discusses the system implementation of AttributeTrust and finally, Section 4.3.4 presents the calculation of confidence metrics.

4.3.1 Desirable Properties

Before we present the model, it is important to understand some of the properties that are desirable in such a model. These properties will serve as a guideline as to how AttributeTrust is expected to behave.

1. Each entity should be able to compute confidence values using the model in reasonable time.

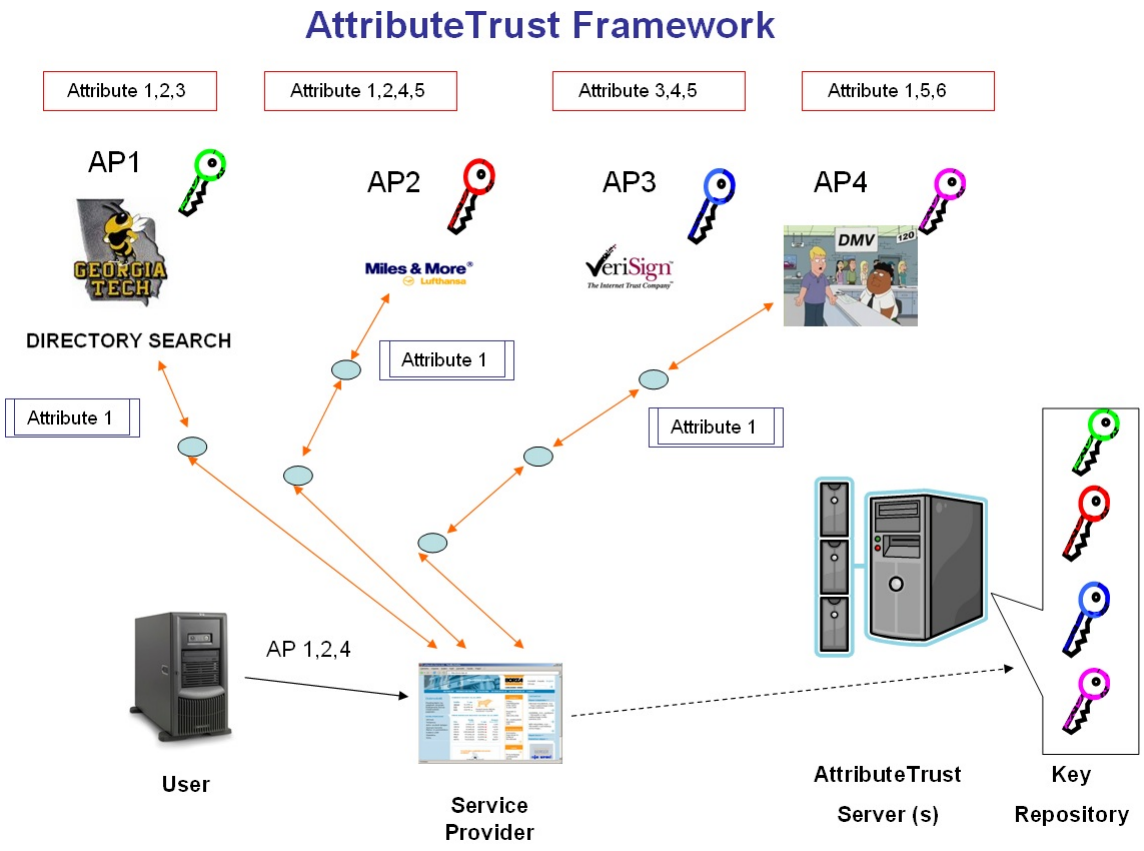

Figure 2: AttributeTrust framework.

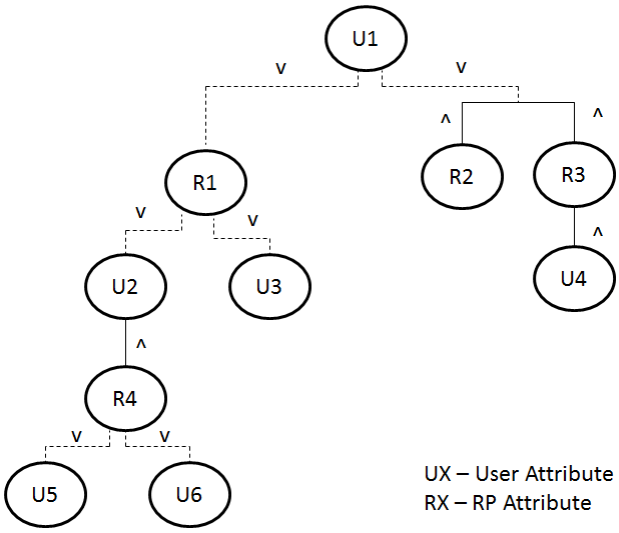

Figure 3: Attribute release policy tree.

2. The output of the model should be a metric that can be directly used in making authorization decisions.

3. Evaluating the metric with only partial information should still give meaningful results (albeit with a lesser accuracy).

4. The model should be robust against common attacks on reputation systems.

5. The metric should degrade gracefully in the event of an intensified attack.

6. There should be disincentive for an honest user to abuse the reputation system.

4.3.2 Model Formulation

The system can be viewed as a weighted directed graph $G = (V, E)$, such that $E \subseteq [V]^2$. Each vertex (node) V represents an entity; every edge E represents a confidence relationship. The entities in the system can be users, APs or RPs. The edges are directed and weights on the edges are not necessarily symmetric i.e. for two entities a and b, as confidence in b can be higher or lower then bs confidence in a. Establishing this directed edge relation is optional for the entities. Nodes in the system representing RPs and users are weighted and the node weight calculation

is explained later in this section. Nodes representing the APs do not have a node weight. The order of graph $\mid G \mid$ is defined by the number of entities in the system. G is irregular and incomplete. Edges and nodes are created dynamically in the graph, so $\mid E \mid$ and $\mid V \mid$ increases with time but for calculating the node and edge weights (to construct a path as explained in Section 4.3.4), we will consider a snapshot of the graph, which is static.

The node weight is represented by ω . The node weight ω_i for node i is defined as

$$\omega_i = d_i \times \overline{C_i} \tag{1}$$

$$\overline{C_i} = \sum_{i=1}^{n} C_{k \to i}, \tag{2}$$

where n is the number of incoming neighbors of node i and
d_i is the in-degree of node i.

The node weight is a product of the in-degree and the average confidence value on the edges, which is simply equal to the sum of the confidence values. A high in-degree means a lot of nodes have expressed confidence in node i. A high average confidence value means that node i's peers have high confidence in it. So the node weight represents the communitys confidence in node i. The edge relation value $C_{k \to i}$ represents the confidence node k expresses in node i. It is a nonbinary value between [0,1]. When an entity enters the system initially, its node and edge weight are zero because it is not connected to any other entity. The node weight increases when edge relationships are established. A default node weight of zero is helpful in reducing the impact of certain types of attacks. This is discussed in more detail in Section 4.4.

The edge relation between nodes representing an AP and a user is different from what is described above. These edges are always directed from the user to the AP. Instead of a single confidence value, the weight of this edge is an array where each array element represents a confidence value for an individual attribute.

4.3.3 System Implementation

In the system, each entity has a unique pseudonym. Each entity has a public-private key pair. The public keys of the APs are stored in the system in such a way that any entity can access them[2]. Entities interact with each other in the process of acquiring or providing service. They provide feedback in the form of confidence scores. The node weight is proportional to the in-degree. If an entity wants to provide feedback to its peer entity after a transaction, it should fulfill two requirements i) have an irrefutable proof that the transaction occurred and ii) have a node weight greater than the participation threshold p .

The participation threshold forces the entity to behave in an honest manner for some time and earn its 'reputation' before it can participate in the feedback process. On the other hand, to establish a user $\rightarrow$ AP relation edge the user should satisfy requirement iii) instead of requirement i); iii) either the user has attributes certified by the AP or the user has performed a transaction in which it accepted attributes certified by the AP. The weight of a node is dependent only on the edges terminating in it, so the weight of each node can be calculated independently. This helps in eliminating oscillations in weight calculation. These weights can be calculated in multiple ways. A central server can be the easiest implementation but it also makes the central server a lucrative target for attacks. A performance problem can arise by a very large number of nodes simultaneously querying the server for node weights. Alternately, a group of servers can be used to calculate the node weights. Each server calculates the weights for one group of entities with overlap, so that each entitys node weight is calculated by more than one server. This provides redundancy to

[2]In the case of an IdP's identity, knowing its public key reliably is equivalent to trusting the IdP. Trust brokering is performed by the CA through a complex trust hierarchy; hence establishing trust through trust chains is difficult. In the case of an AP, knowing the public key of an AP (trust brokered by the AttributeTrust system) is different from trusting the AP (which is based on the AP's reputation). Since there is a single trust brokering entity, it is easy to put all the AP's public keys at one place, the AP's still have to earn a high reputation to be trusted.

resist failures and attacks. As another alternative, entities can calculate node weights where each entity is responsible for a small group of nodes and each nodes weight is calculated by multiple entities [53].

The edge weights are stored locally with each entity and used during the construction of confidence paths (explained in Section 4.3.4).

4.3.4 Metric Calculation

Referring to Figure 4, consider a scenario where node j is a service provider (acting as a relying party) and node i wants to acquire a service. During their negotiation, node i provides some attribute issued by AP1 to node j. In order to calculate its confidence in this attribute, node j finds all the paths from itself to AP1 of maximum length L. Using this parameter, the relying party places an upper limit to the number of redirections in the transitive trust chain. The Relying Party may not be comfortable accepting attributes from very far-off nodes, hence this parameter is useful. This parameter also helps the algorithm to converge in reasonable time. Node j asks all its peers if they have a confidence edge leading to AP1. The peers in turn repeat this process until either they find a path to AP1 or the path length exceeds L. j finds two paths viz $j \rightarrow i \rightarrow 4 \rightarrow AP1$ and $j \rightarrow 5 \rightarrow 8 \rightarrow 9 \rightarrow AP1$. Upon receiving values along all the paths, node j converts them into node disjoint paths. For paths that have parallel sub-paths, final confidence value is calculated (as explained later in this section) for each sub-path. The sub-path with the highest value is considered. For example, in Figure 5 we calculate confidence values of $j \rightarrow 7 \rightarrow 1 \rightarrow i$ and $j \rightarrow 2 \rightarrow i$ and choose the higher value of the two. For calculating the final confidence values, we define two functions concatenation and aggregation.

Concatenation (Θ) is used to find the confidence for one complete trust path. We multiply the confidence $C_{t \rightarrow t+1}$ by ω_t and repeat the process for each of the nodes in the path (t represents intermediate node). For this, we consider ω_j as 1, instead of

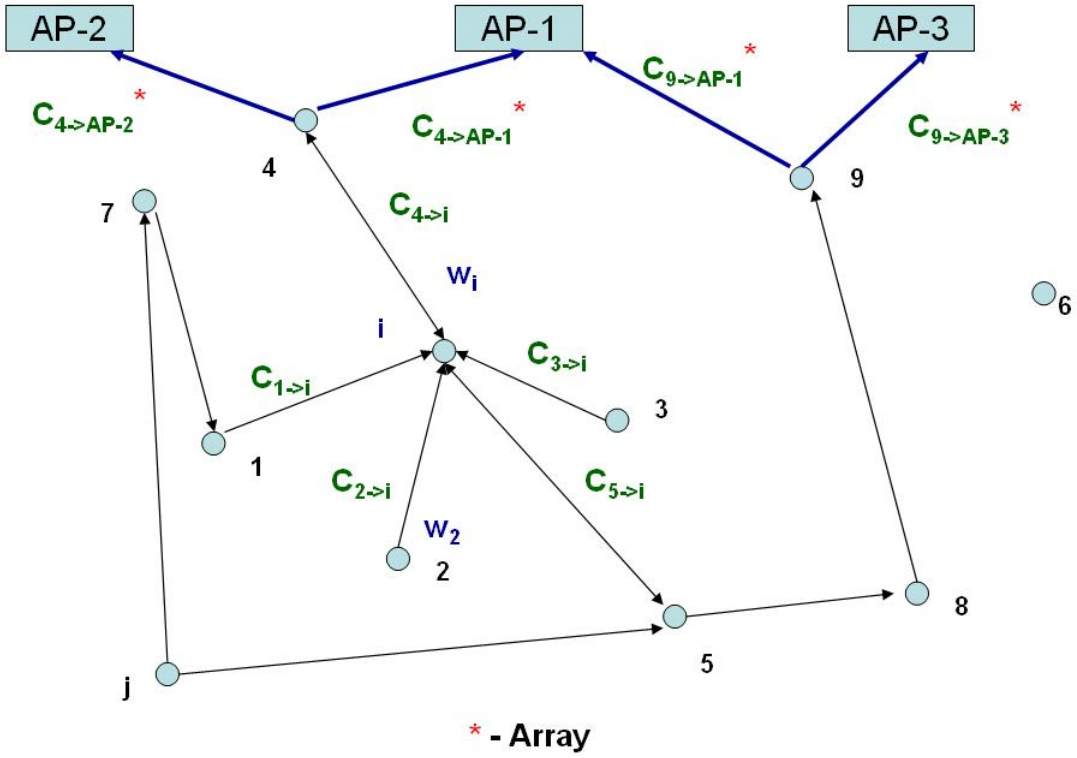

Figure 4: Metric calculation.

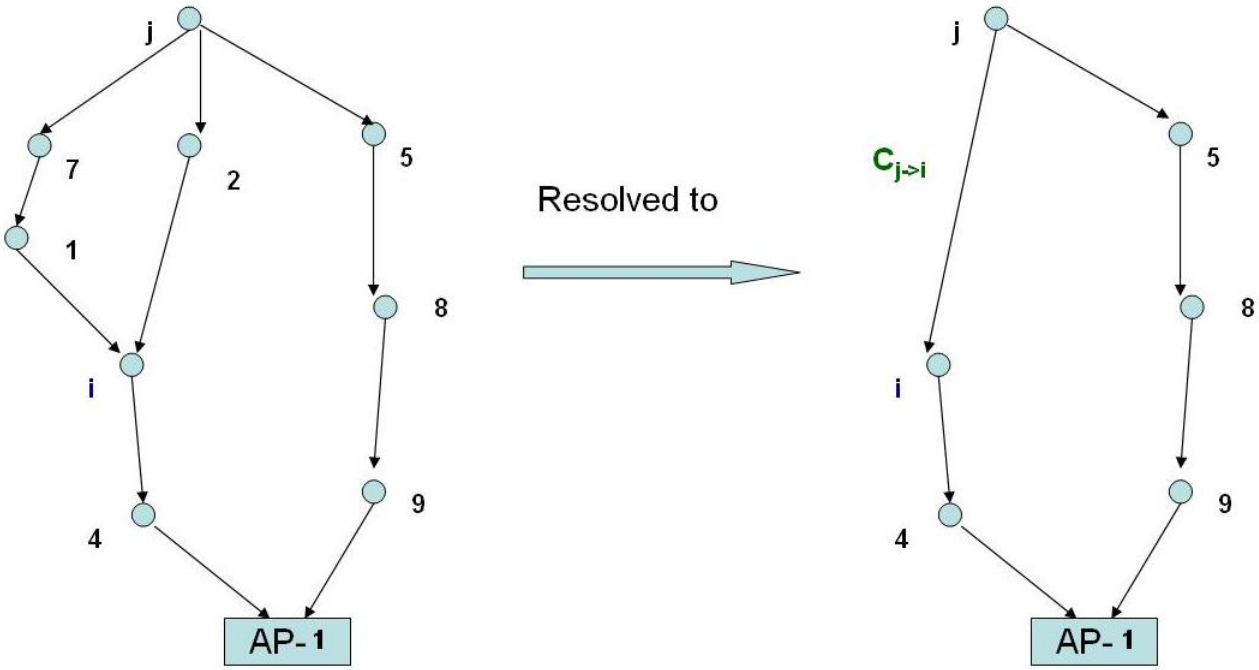

Figure 5: Node joint path resolution.

its global node weight, because j will have complete faith in the confidence value he gave to his peer.

$$\Theta = \omega_t \times C_{t \to t+1} \tag{3}$$

Aggregation **(Σ)** is used to find the final confidence value (Γ) considering all the node disjoint paths. There are many aggregation functions possible. We present four popularly used aggregation strategies and examine the effect of a slandering attack on AttributeTrust for these strategies in section 4.4.4.

1. A conservative strategy is to choose the confidence value of the path with the minimum value.

2. Another strategy is to find the average confidence value of all the available paths.

3. Another popular strategy requires that at least 3 confidence paths are available. They are sorted in order of their confidence values and the median confidence value is chosen.

4. In the last strategy, known as the additive strategy, the final confidence value is the sum of the confidence values of all the paths; each newly created path will increase the final confidence value.

For each node disjoint path from node j to AP1, we apply the concatenation function to calculate the path wise confidence values and then we apply the aggregation function including all the paths to calculate the final confidence value Γ. The AP is considered sufficiently trusted by the RP if the value of Γ is greater than some user defined threshold. A suitable default value will be set but the user will have an option to change this value depending on the sensitivity of the transaction. From the usability point of view, the system should have a default threshold value in place and the user should interact with the system only while providing feedback to its

peer entity. To further improve usability, this feedback can be represented as several discrete levels like completely trusted, highly trusted, average, marginally trusted and untrusted.

4.4 Attack Resistance

In this section, we take a look at some of the common attacks and abuses against reputation systems and how the proposed model defends against them.

4.4.1 Whitewashing

When the reputation of a user in the system becomes very low, he discards his pseudonym and reenters the system with a new pseudonym. Although acquiring a pseudonym in the system is free, we provide a disincentive for the user to do that. When a new user enters the system, he is not connected with any other node and his node weight is zero. Low reputation equates to a low node weight in the system. If a user reenters the system, he has a zero weight which is the minimum, so there is little incentive for him to acquire a new pseudonym.

4.4.2 Discrimination

Some users behave well with most of the users but discriminate against some of them. We address this problem by using a link drop threshold μ. Suppose user i constantly discriminates against user j, the confidence value $C_{j\ rightarrowi}$ will keep on dropping. We define a threshold below which user j will not consider paths via user i for calculating confidence metric irrespective of ω_i . This way user j can eliminate discriminating nodes from his confidence metric calculation.

4.4.3 Traitors

In this attack, a malicious user behaves honestly for some time and then tries to milk his reputation by cheating. This reputation building and milking cycle continues. During the reputation milking phase, the user cheats on a number of transactions

continuously until his reputation is very low; and then either enters the reputation building stage or reenters the system with a new pseudonym. As an extension, AttributeTrust can provide a users history of recent transactions, which the other party may use to call off a negotiation if the user had a majority of his recent transactions failed. This way AttributeTrust cuts short the milking stage of a traitor.

4.4.4 Slandering

A malicious user may try to malign the reputation of a reputed AP by launching a slandering attack against the AP. To launch this attack, the malicious user M creates a relation edge from himself to AP1 as shown in Figure 6. He can easily create this edge by performing a transaction with a user and accepting attributes from AP1. M then gives arbitrarily low confidence values to attributes provided by AP1. The effect of a slandering attack by M by adding a malicious path P_M (to the honest paths P_4 and P_9) is shown in Figure 7. Similarly a group of colluding users can create numerous paths from user to AP1. The effect of this depends on the choice of aggregation strategy (different strategies presented in Section 4.3.4) and a comparison of different strategies is presented more formally below.

The aim of adversaries in this case is to reduce the APs reputation by as much as possible. They try to achieve this by creating paths of very low trust values. Here, we present the cost of launching such an attack and a lower bound on the confidence value for different aggregation schemes. Let, m and ρ be the minimum number and actual number of paths respectively, (assuming, $\rho \geq m$), T be the confidence value of a path i, ϵ be the confidence value of one malicious path, l be the lower bound on the confidence value in the presence of malicious entities and κ be the number of malicious paths created. Further, let, ξ be the cost of launching the attack and χ denote the cost of an edge attack[3], as described in [85]. Note: $\epsilon \cong 0$ $s.t.$ $\kappa.\epsilon \approx \epsilon$. In

[3]The cost of launching an edge attack is assumed to be uniform across all nodes as in [57] and [85].

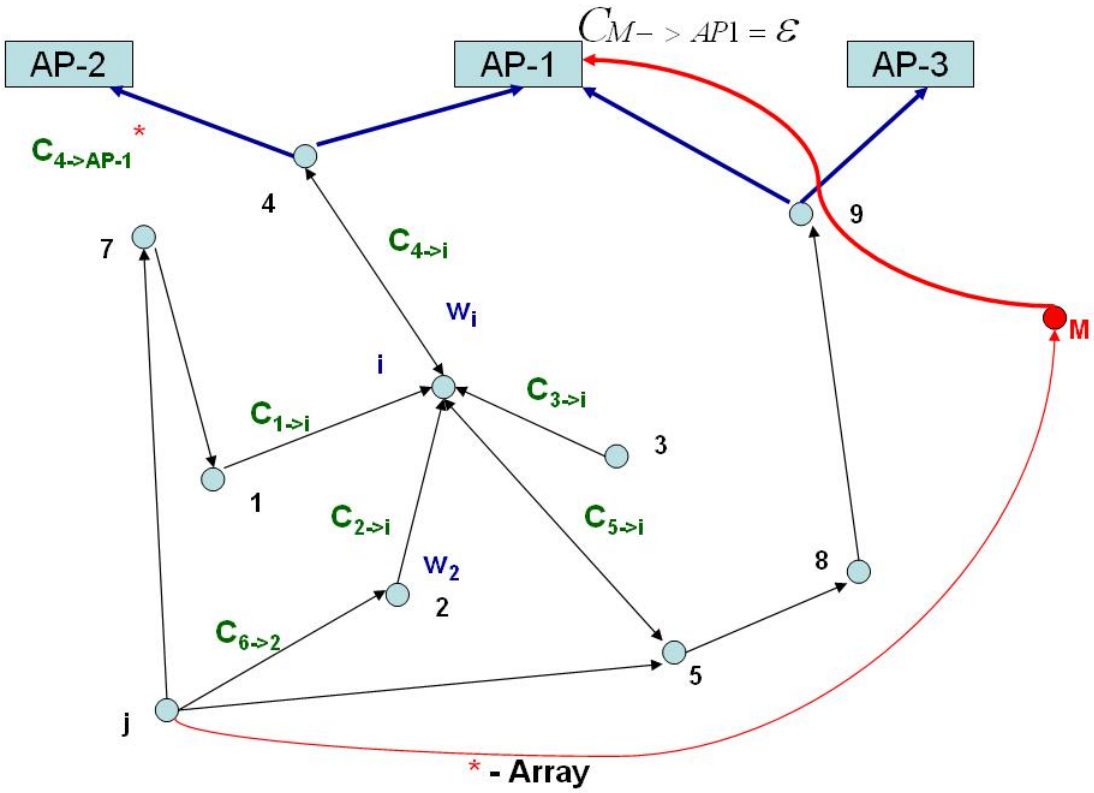

Figure 6: Slandering attack.

Minimum Path –	P_M	(High Impact)
Average of Paths -	$(P_4 + P_9 + P_M)/3$	(Medium Impact)
Discard Extremes –	P_9 if $P_4 > P_9 > P_M$	(Low Impact)
Summation over paths -	$(P_4 + P_9 + P_M)$	(No Impact)

Figure 7: Effect of a slandering attack.

the following, i to iv refer to the aggregation strategies discusses in Section 4.3.4.

Theorem 1 *Using strategy (i), the final confidence value can be made arbitrarily low with a cost of* χ.

Proof If the malicious entity creates a single malicious path with confidence value ϵ, then $l = \epsilon$ and $\xi = \chi$.

Theorem 2 *Using strategy (ii), the malicious entity's ability to bring down the final confidence value is limited by its ability to successfully attack disconnected nodes i.e. by the value of* κ.

Proof To find the average, the summation of the confidence value of paths remain almost constant ($\kappa.\epsilon \approx \epsilon$) but the final confidence value decreases with increasing κ, $\sum_{i=1}^{\rho} T_i = P$, $l = \frac{P+\epsilon}{\rho+\kappa}$ and the cost is $\xi = \kappa.\chi$.

Theorem 3 *Using strategy (iii), the final confidence value can be brought arbitrarily low if* $\kappa > \rho$. If $\kappa < \rho$, then the final confidence value is at least P_l, where P_l is the confidence value of lowest value honest path be given as $P_l = \sum_{lowest-value-path} T$.

Proof If $\kappa < \rho$, then in the worse case, the median path is the lowest value path and $l = P_l$. On the other hand, if $\kappa > \rho$ then the median path will be one of the malicious paths and $l = \epsilon$. The cost in either case is $\xi = \kappa.\chi$.

Theorem 4 *Using strategy (iv), the malicious entities cannot lower the final confidence value irrespective of their ability to perform edge attacks.*

Proof In the additive strategy, each new confidence path can only increase the final confidence value, so for κ malicious paths, $l = P + \epsilon$ and the cost is $\xi = \kappa.\chi$. Note that $P + \epsilon > P$.

4.4.5 Self-promotion via Sybil Attack

As an advanced attack, M can create a malicious attribute provider AP-M, create a number of Sybil nodes [41] and try to increase the confidence value of AP-M directly by asserting high confidence values to attributes provided by AP-M. M can further increase AP-M's confidence indirectly by asserting high confidence values to AP-M through all the Sybil nodes. We use a novel way to counter this problem based on the attack graph density.

On the system graph describes in Section 4.3.2, Let $\delta(G)$, $d(G)$ and $\Delta(G)$ represent the minimal, average and maximal degree of the graph. Then the following relation holds true

$$\delta(G) \leq d(G) \leq \Delta(G) \tag{4}$$

We define three types of nodes on the graph. An honest node is one that has attributes issues from reputed APs and has a good intention, a malicious node that also has attributes issues by reputed APs but has a bad intention and a Sybil node which does not have attributes issues from reputed APs and has a bad intent.

Honest nodes and malicious nodes exist as fast mixing sub-groups on the graph, each forming a subgraph G' of high degree but the sub graphs are connected by only a few edges creating a low degree between the sub-groups [89] [1]. This is shown in Figure 8. This is similar to the graph property mentioned in [89]. We call them the honest region and the malicious region on the graph. So within the sub-graphs the degree is very high i.e. closer to $\Delta(G)$ and between the sub-graphs it is very low i.e. closer to $\delta(G)$. This is true because honest nodes interact a lot among themselves and Sybil nodes interact among themselves. For a Sybil node it is more difficult to perform a transaction and receive a confidence score from an honest node. This is true because the very few honest nodes will accept the Sybil nodes attributes because they are not certified by reputed APs. On the other hand, the Sybil nodes can arbitrarily

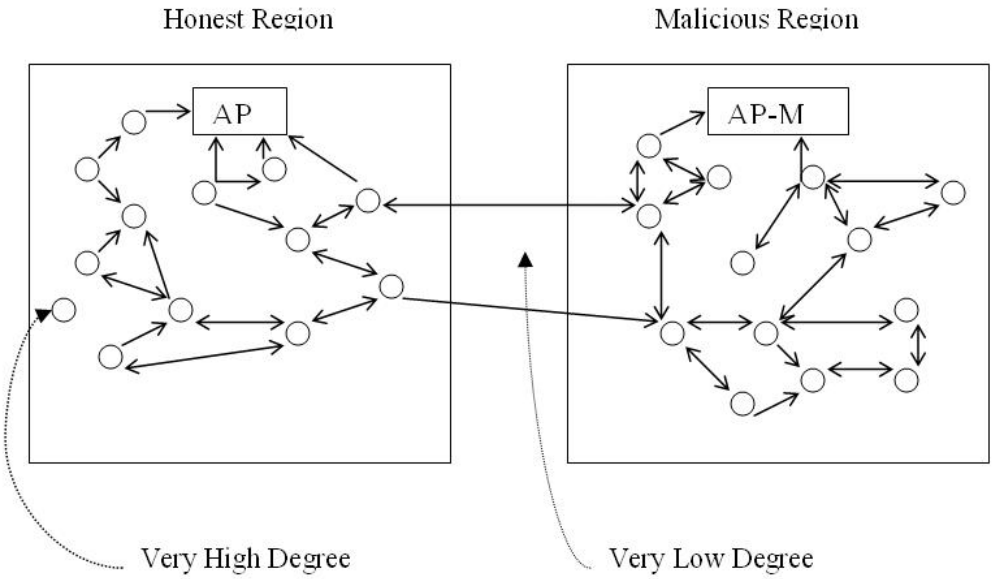

Figure 8: Low degree between sub-graphs.

provide high confidence scores to each other.

Now, we slightly modify the method of calculating Γ as defined in Section 4.3.3. When the Relying Party finds all the node disjoint paths to the AP, it will calculate Γ for the complete graph G (represented as (Γ, G)) as described earlier. Then the RP removes one of the paths to create a sub-graph H(E',V'). H ⊆ G such that E ⊆ E and V ⊆ V. Now the RP calculates Γ for graph H, represented as (Γ, H). The RP repeats this by removing each path once and calculating the resulting Γ. The normalized difference is calculated in a variable λ for each removed path as shown in equation 5.

$$\lambda = \frac{(\Gamma, G) - (\Gamma, H)}{(\Gamma, G)} \tag{5}$$

We examine the change in λ for each removed path. If the RP and AP both are in the honest region, there will be numerous disjoint paths with similar path confidence values. If we remove one path out of n, the drop in (Γ, H) compared to (Γ, G) will be comparable to 1/n. On the other hand, if the RP is in the honest region and the AP

is in the malicious region, there will be very few paths from RP to AP. If we remove a path now, the drop in (Γ,H) compared to (Γ, G) will be much higher. This can be explained by the fact that M and Sybil nodes will jointly try to increase the confidence value for that path[4] artificially to a very high value. If this path is removed, the drop will be very high. This drop can be characterized and compared against a threshold. We call it the Sybil Threshold. If the value of λ is higher than the Sybil Threshold, we say that the confidence value on the path is artificially inflated and remove this path in calculating Γ. This way we reduce the threat of a malicious user inducing a bogus AP in an attempt to subvert the system.

Using the notation from Section 4.4.4, the cost of launching a successful self-promotion attack is $\xi = d.\chi$ s.t. Γ ¿ acceptance threshold and λ ¡ Sybil threshold. Unlike the slandering attack where the malicious entities can mount edge attack on source node itself, a self-promotion attack will be more effective if the malicious nodes target well connected nodes with high node weights. Each confidence value from the source node to the AP-M will have two parts, from the source node to the victim[5] node (denoted asT_{honest}) and from the victim node to the AP-M (denoted as$T_{malicious}$). The malicious nodes can make $T_{malicious} \approx 1$.

Theorem 5 *In the event of a self-promotion attack,* Γ *has an upper bound of* $\sum_{i=1}^{\rho} T_{honest,i}$.

Proof For a particular path i, $C_{source \to AP-M} = T_{honest,i} \times T_{malicious,i}$, since $T_{malicious,i} < l$, $C_{source \to AP-M} < T_{honest,i}$. So for all the available ρ paths, $\Gamma = \sum_{i=1}^{\rho} T_{honest,i}$.

4.5 Discussion

In this section, we discuss how AttributeTrust fulfills some of the model requirements mentioned in Section 4.3.1.

[4]Highly connected malicious node and Sybil nodes will form a single node disjoint path.
[5]Victim nodes are victims of successful edge attack by the malicious entity.

1. To achieve a short aggregation time, the length of each confidence path is bounded by L. Assuming each entity has n peers, the maximum operations to find all the confidence paths are bounded by n^L. After the confidence paths are determined, fetching the node weights and edge confidence values to evaluate the final confidence value should be a trivial task.

2. The output of the metric is a numeric value that can be directly compared against the acceptance threshold to make authorization decisions.

3. If we evaluate the final confidence value with only a subset of paths, we still get a confidence value. Although this value is lower than the actual value, it is still useful.

4. This requirement was discussed in detail in Section 4.4. In a self-promotion attack, the metric output will degrade if malicious users can provide a confidence path from the RP to the malicious AP. As the number of malicious paths increase, the output degrades roportionally. Thus, as the attack is intensified, the metric output degrades gracefully rather than the metric breaking down completely at some point.

5. This requirement was discussed in detail in Section 4.4.1 and 4.4.3.

4.6 Related Work

4.6.1 Attribute Aggregation

If a user has his attributes distributed among multiple providers, these attributes need to be aggregated in a single session and provided to the relying party. Klingenstein [54] looked at this problem in the context of a federation. In his paper, he assumes that the user will authenticate to each IdP and decide which attributes to provide to the SP. In contrast, we propose a policy based approach where the user agent decides which

attributes to disclose. Klingenstein argues that this problem is rarely encountered in open systems, while we formulated our AP model especially for open systems.

Chadwick [38] looked at the problem from the point of view of users privacy and issues in linking the users different pseudonyms in different attribute authorities (AA). He proposed a protocol for linking the AAs, SP and the user completely under the users control but the protocol requires intensive participation from the user to authenticate with the AA and linking the AAs to aggregate attributes for each session. AttributeTrust uses a policy based approach where the attributes are aggregated either by the user (automatically through the client program) or the RP. The authentication mechanism for gathering attributes in our framework is also more sophisticated.

Chivers [39] looked at user attribute aggregation as a method of constructing a users profile via data mining. His method limits the disclosure of attributes to Service Providers by classifying attribute types to protect user privacy. One SP can request attributes from only one type. To classify the attributes, this method assumes that the user has a very good idea of what attributes a SP will require for a particular authorization. In contrast, AttributeTrust uses the minimum information disclosure principle to achieve the same goal without requiring this assumption. Also the user controls the aggregation process at all times, so the user privacy is protected. On the other hand, our model uses an opaque token for delegation which is similar to Chivers recommendation.

4.6.2 Trust Negotiation

Winsborough et al., [87] present one of the earlier works on negotiating disclosure of sensitive credentials. They proposed eager and parsimonious strategies and compare their performance.

Squicciarini, et al., [83] presented a framework for negotiating release of sensitive

attributes. They discuss several interoperable negotiation strategies for improving privacy. This work is based on the assumption of a standard IdP and we extended these concepts complementing the scope and functionality of the AP. Seamons et al., argued that access control policies protecting sensitive credentials are themselves sensitive and their disclosure should be limited [79].

4.6.3 Confidence measurements via Reputation systems

The Eigentrust algorithm was proposed by Kamvar et al., [53]. Their approach is based on transitive trust. They solve the problem of colluding adversaries by assuming that there are a small number of pre-trusted peers whom the model trusts unconditionally (similar to [57]). We argue that pre-trusted peers may not be available for other applications. Although our work is also based on the notion of transitive trust, it differs from their model because our model does not assume any seed of pre-trusted peers in the system. We solve the problem of colluding adversaries by the appropriate choice of aggregation strategy. Additionally, our work extends previous research by including attribute aggregation and trust negotiation. Hoffman et al., [48] did a survey of attack and defense techniques for reputation systems. It covers a broad range of attacks and defense techniques. Douceur [41] introduces the Sybil attack, which is a rampant problem in reputation systems. He argues that without a centralized identity authority, it is not possible to control the Sybil attack. AttributeTrust employs a novel method for limiting the impact of Sybil attack by exploiting the property of varying degrees in the system graph. Although we do not eliminate Sybil entities from the system, we effectively isolate them during evaluation of our confidence metric.

Yu, et al., proposed SybilGuard [89] for defending social networks against Sybil attacks. Their model has few attack edges between the honest and Sybil part of the graph. Each edge is a human established trustrelationship among nodes with out of band verification. Our model similarly relies on varying degrees of the system

graph and we believe that this property exists in networks where establishing an edge relationship requires a strong credential. However, we argue that out of band verification using phone for each trust relationship assumes high involvement of the user. In AttributeTrust, we use trusted attributes as credentials to create the relationship edge and the human interaction is minimal in the form of providing feedback for successfully completed transactions. Moreover, SybilGuard uses random routes method to provide an upper bound on the number of Sybil nodes as against our method of eliminating the node disjoint paths from confidence calculations. Another difference between the two models is that in SybilGuard, all the edges have equal weights. In AttributeTrust, edge weights (confidence values) and node weights are central to calculating the final confidence values. Another Sybil attack resistant recommendation system was proposed by Seigneur et al., where they implemented their system in email anti-spam settings [80].

Xiong, et al., proposed PeerTrust [88] a reputation based trust model for P2P electronic communities. Their trust model has five factors some of which are fixed and some of which are adaptive in nature. For countering adversary collusion, each node maintains a similarity context with every peer by giving a weight to a peer by seeing how similar their feedback was to all the other common peers. While their model produces good results, this may lead to the problem of Groupthink, where new participants will not give a poor feedback to a node which received a good feedback from his peers for fear of reducing its own weight.

Guha, et al., [46] propose a model for propagation of trust in recommendation systems. They propose a model based on transitivity of trust and distrust. Their model predicts trust between any two entities in the system even with few expressed trust relationships per node. However, certain contributions of AttributeTrust like fine grained confidence for each attribute, scaling a users opinion by his node weight are not in the scope of their work.

Reiter, et al., [72] and Beth, et al., [33] have proposed authentication models based on transitive trust. Their notion of transitive trust is verifiability of target users public keys which is quite different from ours. Levien, et al., [57] proposed attack resistant trust metrics based on a notion similar to [72] which assumes a seed of trusted peers.

Buchegger, et al., [36] used a Bayesian reputation system model to mitigate slandering attack by eliminating transitive information that deviates substantially from direct information. Our model mitigates the slandering attack by choice of appropriate aggregation strategy.

CHAPTER V

DYNAMIC AUTHORIZATION

Policy-based authorization systems are becoming more common as information systems become larger and more complex. In these systems, to authorize a requester to access a particular resource, the authorization system must verify that the policy authorizes the access. The overall authorization policy may consist of a number of policy groups, where each group consists of policies defined by different entities. Each policy contains a number of authorization rules. The access request is evaluated against these policies, which may produce conflicting authorization decisions. To resolve these conflicts and to reach a unique decision for the access request at the rule and policy level, rule and policy combination algorithms are used. In the current systems, these rule and policy combination algorithms are defined on a static basis during policy composition, which is not desirable in dynamic systems with fast changing environments. Some other methods to combine authorization policies are discussed in [60], [58].

In this chapter, we motivate the need for changing the rule and policy combination algorithms dynamically based on contextual information. We propose a framework that supports this functionality and also eliminates the need to recompose policies if the owner decides to change the combination algorithm. It provides a novel method to dynamically add and remove specialized policies, while retaining the clarity and modularity in the policies. The proposed framework also provides a mechanism to reduce the set of potential target matches, thereby increasing the efficiency of the evaluation mechanism. We developed a prototype system to demonstrate the usefulness of this framework by extending some basic capabilities of the XACML policy

language. We implemented these enhancements by adding two specialized modules and several new combination algorithms to the Sun XACML engine.

5.1 Introduction

As information systems become more complex and distributed in nature, system administrators and users need authorization systems which can help them share their resources, data and applications with a large number of users without compromising security and privacy. Although traditional authorization systems address the basic problem of granting access to only authorized individuals, they do not provide a number of desired features of modern authorization systems. These include 1) easily changing authorization based on accessor roles, group memberships, institutional affiliations, location etc., 2) multiple authorities jointly making authorization decision, 3) dynamically changing authorization based on accessor attributes, and 4) GUI-based general purpose tools for description and management of authorization rules. Some traditional authorization systems provide some of these functions on an ad-hoc basis. Although policies have always been part of authorization systems, they were mostly buried in other functional code and hence were difficult to compose and analyze.

Modern policy-based authorization systems provide most of these features. They have a separate policy module that can be queried to make authorization decisions. This module makes decisions taking into consideration all applicable policies for a particular access request. These policies may be defined by multiple authorities. The policies may have different or even conflicting authorization decisions for the same access request. Policy languages use policy combination algorithms (PCA) to resolve such conflicts. These algorithms take the authorization decision from each policy as input and apply some standard logic to come up with a final decision[1].

These PCAs are currently chosen at the time of policy composition and hence

[1]For efficiency reasons, policy engines only evaluate policies until they reach a final decision based on the combination algorithm.

they are static. In highly dynamic environments, this is not desirable and there may be a need to select these PCAs dynamically. In this case, it will be useful to have a mechanism to select a suitable PCA based on the dynamic contextual information available to the system. More discussion on this issue along with a motivating scenario is presented in Section 5.3.

PCAs used in current systems are also very restricted. There are a number of conflict resolution logics in general purpose computing which are not expressible as PCAs in authorization languages. Examples of these logics include hierarchy-based resolution, priority-based resolution, taking a simple majority vote, and taking a weighted majority vote. There is a need to include algorithms such as these as PCAs in authorization languages to provide more functionality and flexibility in defining policies.

Having a context-aware authorization system also provides the capability to define different policies for different contexts. These contexts can be distinguished by contextual data or environmental attributes. In this case, the policies will be modular making them easy to comprehend and analyze. Without the ability to choose the applicable policies based on contextual information, the policy composer is forced to duplicate each access control rule with and without the contextual information in the same policy. Although the same access control decision can be achieved in both approaches, the latter makes it difficult to analyze the policies and the effect of making changes to them. Also if policies are chosen dynamically, only a small set of rules will be evaluated for their applicability for this request. This reduces the number of matches with potential policy targets thereby lowering computation time.

Another advantage of using context-aware authorization is that a specialized policy created for some specific purpose can be added and removed from consideration dynamically without changing the existing policies. This is especially useful for systems that have to adhere to certain temporary authorization requirements which

require special authorization rules. This is also useful in cases where the specialized policy is composed by some entity other than the one who usually creates and maintains authorization policies.

5.2 Attribute-based Authorization Systems

In this section, we first introduce the basic constructs of attribute-based policy languages. We then describe some basic concepts of attribute-based authorization systems, define attribute-based policies, and policy combination algorithms used in conflict resolution.

5.2.1 Brief Introduction to Policy Languages

In this sub-section, we introduce the basic elements of attribute-based authorization policy languages. Although here we use eXtensible Access Control Markup Language (XACML) as an example to introduce the primary elements, these elements are similar in other policy languages as well.

XACML is an OASIS standard that describes a policy language for representing authorization policies and an access control decision request/response language [5]. XACML is based on XML. It describes general access control requirements while allowing for extensions for defining new functions, data types and combination logics. The language has syntax for defining authorization policies and building a request/response to validate authorization requests against the policies. The response contains one of the four possible outcomes of policy evaluation - Permit, Deny, Indeterminate (an error occurred or some required value was missing, so a decision cannot be made) or Not Applicable (the request can't be answered by this service).

XACML has a Policy Enforcement Point (PEP) that actually protects the resource and a Policy Decision Point (PDP) that evaluates the access request against the policies. The PEP receives the access request from the requesting user and forwards it to the PDP which makes the decision in consultation with the policies. If the access

is allowed, the PEP release the resource to the requesting user. The main components of a XACML policy are described below:

Policy - An XACML policy contains a set of rules with the subject and environment attributes, resources and corresponding actions. If multiple rules are applicable to a particular request, then the rule combination algorithm (RCA) combines the rules and resoles any conflict in their decisions. XACML supports the following RCA's - Deny-overrides (Ordered and Unordered),Permit-overrides (Ordered and Unordered), and First-applicable.

Policy Set - A policy set is a container which contains other policies or policy set. One or more of these policies or policy sets may be applicable to a particular access request. If more than one are applicable, then the Policy Combination Algorithms (PCA) are used to combine the policies and resolve any conflicts in their decisions. XACML supports the following PCA's - Deny-overrides (Ordered and Unordered),Permit-overrides (Ordered and Unordered), First-applicable, and Only-one-applicable.

Target - A Target is basically a set of conditions for the Subject, Resource and Action that must be met for a Policy Set, Policy or Rule to apply to a given request.

Rule - The rule is the core representation of the access control logic with the subject, resource, action and environment fields. It is a boolean function, which evaluates to true if the subject, resource, action and environment fields in the request matches with the fields in the rule.

5.2.2 Authorization Policy

In an attribute-based system, objects are protected by administrator (or object owner) defined policies. These policies define a set of verifiable attributes (with pre-defined values) against each resource for a set of privileges. These attributes are either the characteristics of the user or the environment. These attributes must be presented

to the authorization module and verified by it in order to authorize the accessing user to access the requested object with specific privileges. Since the attributes have to be verifiable, they have to be certified by some entity which is trusted by the authorization module.

An attribute-based authorization policy is formally defined below.

Definition 1 : Let $\mathcal{SA}$, $\mathcal{RA}$ and $\mathcal{EA}$ represent the Subject, Resource and Environmental attributes respectively, each of which is well defined set of finite cardinality, given as $\mathcal{SA} = \{sa_1, sa_2.....sa_l\}$, $\mathcal{RA} = \{ra_1, ra_2.....ra_m\}$ and $\mathcal{EA} = \{ea_1, ea_2.....ea_n\}$. These attributes can take values $val\_sa_i \subseteq dom(sa_i)(1 < i < l)$, $val\_ra_j \subseteq dom(ra_j)(1 < j < m)$ and $val\_ea_k \subseteq dom(ea_k)(1 < k < n)$.

Attributes can be of two types, one which can take distinct and unconnected values (for e.g. 'role'='doctor' or 'role'='nurse') and another type which can take a single or range of values (for e.g. 'time' is between t_1 and t_2 or 'age' ≤ 21). In the latter case, the values that an attribute can take are connected. Without loss of generality, we define the latter group as attributes which can take either a single value or a range of values. For example, for a range of sa_j, the domain and values are defined as follows:

Attribute Type 1 -

$dom(sa_j) = [sa_j\_val_1, sa_j\_val_2...sa_j\_val_n]$, $val\_sa_j \in dom(sa_j)$;

Attribute Type 2 -

$dom(sa_j) = [low, high]$, $val\_sa_j = [low', high'] \subseteq dom(sa_j)$; where, $(low' \geq low)$ and $(high' \leq high)$. If val_sa_j takes a distinct value in $[low, high]$, then $low' = high'$.

Definition 2 : Let Action define a set of actions which a subject can execute on resources. $\mathcal{ACT} = \{act_1, act_2.....act_p\}$. For example, the set of actions on a file can be {read, write, delete, append, execute}. Let $\mathcal{D}$ be the set of decisions that can result as a response to a predicate evaluating to true. $\mathcal{D} = \{d_1, d_2.....d_q\}$.

Definition 3 : An access request ($\mathcal{AR}$) is a tuple of the form $< s, r, a >$, where $s \subseteq \{\mathcal{SA}, \mathcal{EA}\}$, $r \subseteq \{\mathcal{RA}\}$ and $a \subseteq \{\mathcal{ACT}\}$. It represents that s is requesting to

access r with rights a. A Rule $\mathcal{R}$ has the same format but defines the set s required to access r with rights a.

Definition 4 : A policy is a list of rules given as $\mathcal{P} = (\oplus, < \mathcal{R}_1, \mathcal{R}_2 ..... \mathcal{R}_s >)$. $\oplus$ is a combination function, which combines the rules to produce a single decision for the policy.

Definition 5 : A Policy Set (PS) is a container which contains a list of policies. It may also contain other policy sets. It is given as $\mathcal{PS} = (\oslash, < \mathcal{PS}_1, \mathcal{PS}_2, ... \mathcal{PS}_i >)$. Each $\mathcal{PS}_t$ represents either a policy set or a single policy[2]. $\oslash$ is a combination function, which combines all the policy sets. This combination function is used to combine policies and policy sets and has no direct relation with the rule combination algorithm.

Conceptually, a policy is a deliberate plan to implement authorization to a particular resource or group of resources. A rule is a component of the policy that defines a specific authorization predicate. A policy set is a container that contains a number of logically connected policies. In a multi-authority setting where the authorization policies for a particular resource are defined by a number of entities, all policies for that particular resource will form a logical policy set. For example, at a university, the firewall policies to protect a lab computer may be a combination of the policy defined centrally by the office of information technology, a specific department policy, a lab firewall policy, and the administrator defined policy for that computer. A policy set encompasses all of these policies. The policies can be defined in a number of policy description languages. Each has its advantages and disadvantages. In describing the policies, we will use the syntax and structure of XACML [5], which is an OASIS standard. XACML is an attribute-based policy description language and is used for implementing our prototype system. Although we use XACML for discussion and implementation, the model we present here is generic and can be implemented in

[2]In which case the set has a single policy and no PCA.

other policy languages like P3P [12] or EPAL [43].

5.2.3 Combination Algorithms and Conflict Resolution

In a large system, there may be multiple authorities who specify the authorization policies. As such, there can be multiple groups of policies. When a request is evaluated in the system, the authorization module determines which policy sets apply to the particular request. Then it checks which policies among those groups and which rules among those policies are applicable to the request. There can be multiple policy sets and multiple policies in each set applicable to a single access request. Even within each policy there can be multiple rules which apply to the access request. These rules and policies can have a different or even conflicting decision for the request. As such, a mechanism is needed to resolve these conflicts. Policy languages have some rule combination algorithms (RCAs), which evaluate the applicable rules based on the logic of the algorithm and resolve any conflict in their decisions.

Definition 6 : In a single policy, $\mathcal{E}(\mathcal{AR}, \mathcal{R}_i) \rightarrow d_i$, where $\mathcal{E}$ represents the evaluation of the i^{th} rule and d_i is the corresponding decision. The set of all the decisions is given as $\mathcal{D}^{Rule} = (< d_1, d_2, ..., d_x >)$. Rule Combination Algorithm (RCA) is defined as $\{\mathcal{RCA}\ \phi\ \mathcal{D}^{Rule}\} \rightarrow d$, where $d\ \epsilon\ \mathcal{D}$. ϕ represents *'applied to'*.

For example, a policy may use 'deny-overrides' as its RCA. In this case, if the algorithm finds even a single rule that denies the access, its final decision is 'deny'; otherwise its decision is 'permit' even if a single rule permits. If none of the rules either 'permit' or 'deny' the access, then the result is 'Not Applicable'.

For combining the policies and policy groups, policy languages have policy combination algorithms (PCAs). These algorithms work on similar logic as the RCAs. Each policy give a single decision for the access request. The PCA combines these decisions into a single decision by using the PCA logic.

Definition 7 : In the final policy list, $\mathcal{E}(\mathcal{AR}, \mathcal{PS}_i) \rightarrow d_i$, where $\mathcal{E}$ represents the

evaluation of the i^{th} policy set and d_i is the corresponding decision. The set of all the decisions is given as $\mathcal{D}^{PS} = \{d_1, d_2, ..., d_x\}$. Policy Combination Algorithm (PCA) is defined as $\{\mathcal{PCA}\ \phi\ \mathcal{D}^{PS}\} \rightarrow d$, where $d\ \epsilon\ \mathcal{D}$.

In the current systems, these RCAs and PCAs are static and are determined at the time of composing the policies.

5.3 Dynamic Conflict Resolution

In the last section, we saw how RCAs and PCAs resolve the conflicts among rules and policies to give a unique decision for an access request. We also noted that, in existing systems, these RCAs and PCAs are chosen at the time of composing the policies and hence do not change. This static composition may not be suitable for highly dynamic environments where there is a need to adapt the policies dynamically. If such a mechanism is available, then it can also serve as an easy tool for the policy composer, if he wishes to change the RCAs and PCAs without recomposing the authorization policies.

Some researchers have proposed static conflict detection and avoidance, arguing that detecting and resolving conflicts in systems with a large number of policies in real time can be a daunting task [90]. We argue that, even though it is a challenging problem, it is a superior approach. Organization policies, regulatory polices, and user policies change regularly. If we perform static conflict analysis, whenever one of the policy changes, new conflicts can arise requiring some party to change their policies. Also, some policies that conflicted before one of the policies changed and were never composed, may now become acceptable. There is no mechanism to reconsider these rejected policies. Also, the static model does not take into account adding and removing specialized and time limited policies to provide flexibility in policy composition and maintenance.

5.3.1 Motivating Scenario

Let us consider a motivating scenario from the health care domain. Alex is a patient who stores his personal health record (PHR) with his health maintenance organization(HMO) called Superior Health Care (SHC). At SHC the patients' PHRs are stored in a repository where the access to the repository is mediated through a proxy. The proxy stores all the authorization policies. The policies may have multiple groups with policies defined by patients like Alex himself, the hospital which created the record, SHC's organizational policies, federal regulatory policies, and so on. When someone tries to access an EMR for a particular patient, the system will consult the applicable policies to check whether this access is allowed. Assume that, in normal circumstances, the policy combination algorithm used is 'deny-overrides', which is a secure and stringent policy. Suppose that Alex wishes to use a more lenient policy in case of an emergency, where he will share his PHR with any accessor who is authorized by at least one of the applicable policies. In this case, he needs to dynamically change his PCA from 'deny-overrides' to 'permit-overrides' whenever there is an emergency and back to 'deny-overrides' once the emergency is over. The traditional method would require him to change his policies twice to achieve this. If Alex want to have several dynamic options, he will have to change his policy description each time such a dynamic change occurs.

In the proposed model, Alex can define all such dynamic conditions as an attribute-based policy and the evaluation of these policies will determine what PCA will be used for the current access request. The model extends this concept to the selection of the RCA dynamically. It is desirable that the user has the ability to define several dynamic conditions simultaneously, need not change his policy descriptions every time one such condition changes, and also need not keep track of the dynamic changes. This is one of the key advantages of using the proposed system. If Alex tries to achieve the same effect in current policy-based systems with static conflict analysis, when an

emergency occurs he will have to recompose his policy with 'permit-overrides' and resolve all conflicts created in the process. When the emergency is over, he will have to recompose his policies with 'deny-overrides' and resolve all conflicts again. He cannot create a special policy for an emergency, because his two policies are inherently contradictory. This puts a heavy burden on the user and also, by definition an emergency comes unexpectedly, therefore Alex cannot be expected to recompose policies when an emergency has already occurred. In current systems, users like Alex do not change their policies on such events. Our novel framework enables users to achieve this with little effort and provides an important new functionality.

5.3.2 Proposed Model

In this section, we present a novel mechanism to dynamically determine the policies applicable to an access request and to evaluate only the applicable policies. In this model, we evaluate the authorization policies in two stages. In the first stage, we determine which policies are applicable to the current access request and we also dynamically determine which PCA will be used to resolve the conflicts in the authorization decisions. In the second stage, we evaluate only the applicable policies using the PCA selected in the first stage.

During stage one, the total applicable policy set (TAPS) is determined by selecting only those policies where at least one of the authorization rules is applicable to the current access request. If $PS_1, PS_2...PS_n$ are the authorization policy sets, then the TAPS for a particular $\mathcal{AR}$ is given as $TAPS = \oslash\{PS_1, PS_2....PS_n\}$.

The combination algorithm $\oslash$ used is *'all-that-apply'*, which is a new rule combination algorithm defined in Section 5.8. The *'all-that-apply'* algorithm has been implemented in our modified XACML engine (see Section 5.5). To evaluate TAPS, all available policy sets are evaluated as explained in Definition 6. If a policy set has at least one rule that applies to the current access request, we include it in the TAPS.

To find an applicable rule, we consider the subject and environment attributes in the access request (which is the set $\{EA \cup SA\}$) along with their boolean relationships. We then match that with the rules in the policy level target. We try to find a rule with the same set $\{EA \cup SA\}$ with the same relationships so that at least one of the attribute combinations matches with those in the $\mathcal{AR}$. EA, SA and RA are specified in Definition 1.

To aid in determining applicable policy sets, we create a meta-policy file called the M-Policy. This file contains one rule for each authorization policy set in the system. This rule is a copy of the policy level target rule included in each set. This rule is a method, in a language such as XACML, to define whether a particular policy is applicable to the given access request and it makes the processing faster. Including it in the M-Policy file has two advantages, namely the processing of the M-Policy file is much faster compared to evaluating the policy level target rule in each file. These rules are optional in XACML. If they are not present, policy evaluation will take longer. Also, we do not use any rule level targets in the XACML policies. As such, we compare the best case performance XACML can offer with our TAPS algorithm. The *'all-that-apply'* algorithm makes it possible to evaluate all target rules at the same place. Each rule in the M-Policy is evaluated (refer to Definition 6). If a rule evaluates to 'permit', it means that the target rule representing the respective policy is true and that policy is applicable. We then include that policy in the TAPS.

To apply the TAPS algorithm to current XACML based systems, we can create an M-Policy file if all the XACML policies in the target system have a policy level target and no overriding rule level target. In systems where either there are no policy level targets or overriding rule level targets are present, an efficient way to implement the TAPS algorithm is to broadly categorize the available policies and use these categories to select the applicable policies. Although this selection will neither be fine-grained nor accurate, it will still improve the performance of the evaluation system because by

using TAPS we can filter out non-applicable policies at an early stage. So, although the performance will not be optimal in this case, it will still be better than the current performance.

The next step in stage one is to determine the applicable PCA (PCA_{apply}) based on a set of environmental attributes, which define the specific conditions under which each of the PCAs is applicable. These environmental attributes essentially define the context of the $\mathcal{AR}$. Some of these attributes might accompany the $\mathcal{AR}$ while others can be provided by an internal or external system entity. We assume that the dynamic decision of which PCA to select is itself based on a policy. Thus, there is a policy set containing the rules governing PCA selection. The PCA rules are defined so that they are mutually exclusive and only one of them is applicable in a particular situation. Although this might seem complex, it is not really so because there are typically a small number of combination algorithms to choose from. This is enforced by using the combination algorithm $\oslash$ *'only-one-applicable'* to choose among the PCAs. *'only-one-applicable'* returns the applicable PCA if one and only one rule evaluates to 'permit'. If zero or more than one rule (and hence the PCA) evaluates to 'permit', then an error code is returned. All rules in the policy set are evaluated and the applicable PCA is selected to be used for resolving conflicts for this access request.

Now in stage two, the final authorization decision is calculated by evaluating the TAPS as $\mathcal{E}(\mathcal{TAPS}) = \{TAPS^{PCA_{apply}}, AR\}\ \phi\ \mathcal{D}^{PS} \rightarrow d$. As defined in Definition 7, in this evaluation, we consider all policies present in the TAPS and evaluate them against the access request $\mathcal{AR}$. The $\oslash$ used in this case is PCA_{apply}, which is calculated in the previous step.

As an example, using this model, Alex can create a PCA selection rule to the effect that if the $\mathcal{E}_{\mathcal{A}} = (\text{'}emergency\text{'} = \text{'}true\text{'})$, then the PCA 'permit-overrides' is used. The effect will be to allow access to anyone who can satisfy at least one of the

applicable policies. On the other hand, in case where $\mathcal{E}_{\mathcal{A}} = (`emergency' = `false')$, PCA 'deny-overrides' can be used. This will limit access to holders of those attribute combinations that are not denied by any policy and are allowed access by at least one applicable policy. Since this evaluation is done during each access request, the PCA will change dynamically whenever there is an emergency.

In addition to providing this novel functionality, our framework proposes the use of TAPS to reduce the policy set to be evaluated for each access request. As shown in Section 5.6, this improves the real time system performance by 4-8 times. Formulation and evaluation of these rules is explained in more detail in Section 5.4.1.

5.4 System Design and Background Modules

In this section, we will first present the system design for a generic implementation of this authorization framework, and then describe some background modules used for building the prototype.

5.4.1 System Design

The proposed system has a two stage authorization process, where in the first stage the applicable policy set and the applicable PCA is determined and in the second stage the applicable policies are evaluated to reach an authorization decision. For the first stage, the policy is created with an index rule for each policy in the TAPS. An index rule is of the form $< \{\mathcal{SA}, \mathcal{RA}, \mathcal{EA}\} : PolicyId >$, where PolicyId is the index id of a particular policy. For example, if policy 'P1234' is applicable to requests in an emergency scenario, then the index rule will be represented as -

$< \{EMT.EMTLicense = `valid'\} : P1234 >$

$< \{CompanyY.Dispatched = `true'\} : P1234 >$

$< \{EMT.Employer = `CompanyY'\} : P1234 >$

The attribute in the index rule is directly provided by an attribute provider (AP)[3]. In this example, the three attributes jointly establish that the EMT's license is valid, he works for company Y and company Y was dispatched to the emergency by the 911 operator. These attributes will be provided by distinct entities. Using them together can establish a complex fact, which cannot be verified by any single entity in the whole system. Note that if an index rule does not contain any attributes i.e. $< * : PolicyId >$, then it is true by default and that policy is always included.

For an access request, the attributes present in the request are compared against the index rules and, in many cases, only a small number of policies will be included in the TAPS. As a result, the policy evaluation stage will be much faster in these cases. The diagram in Figure 9 describes the dynamic authorization process. A similar policy is created with an index rule for each available PCA. Based on the attributes in the index rules, we determine which PCA will be applied to this particular request.

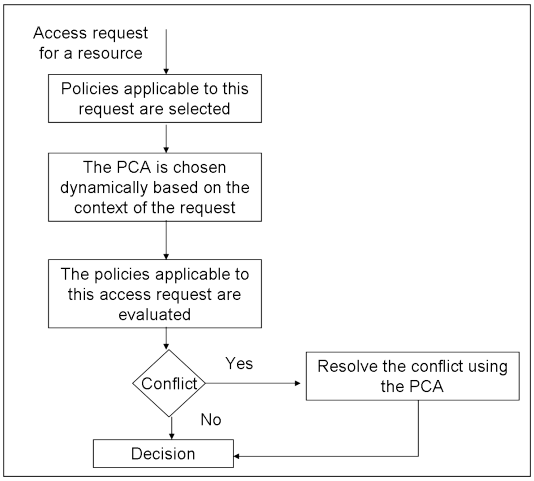

Figure 9: Block diagram of policy evaluation using the proposed framework.

[3]An AP is an entity similar to an identity provider. We define an AP as an entity that can certify certain attribute values for an individual due to its special relationship with the individual. For example, an employer can certify an employee's role in an organization.

5.4.2 Application Scenario

To understand the implication of using context information in the total applicable policy set (TAPS) evaluation and using dynamic PCA selection, let us again consider the previous health care domain scenario. Assume that Alex's HMO where he stores his PHRs has access policies for data based on criteria like data type, membership type, etc. Alex's policies also apply to his PHR, as described earlier. Now Alex, who lives in Atlanta is planning a trip to Florida for a week and he wants his PHR to be accessible to any physician or 'paramedic in Florida' during that week in case he needs medical help. Using our proposed model, he can add a special policy saying $< \{startdate \leq date \leq enddate\} : P2345 >$, where P2345 describes the special permission to 'physicians' in general and 'paramedics in Florida'. Upon evaluating this index rule, Alex's authorization system will compare the current date with the date range in the index rule and will include P2345 during that particular week. Since the proposed model is attribute based, Alex can take advantage of this by adding multiple attribute combinations. Assume that Alex's location can be tracked from his mobile phone, which communicates that to his authorization system over a secure channel. Then Alex can set the index rule as follows : $< \{startdate \leq date \leq enddate\}, \{location = Florida\} : P2345 >$.

This additional attribute will make sure that the lenient PCA is chosen only when he is physically in Florida[4]. Alex's mobile phone is used to provide his location, but the PHR will be primarily be accesses by the paramedics and physicians using their systems. In the event that he has to cancel his trip, his more lenient policy will not be in effect and his information will not be available to any paramedic in Florida. He also has the convenience of setting this rule once and then forgetting about it, irrespective of whether he actually makes the trip or not.

[4]We assume that Alex always carries his mobile phone with him because in essence the service is tracking a device and not Alex himself.

It is important here to note the difference between creating a new access rule in Alex's policy vs. creating an add-on access policy. While the former is possible using the current authorization systems, it will require Alex to modify his policy by adding new access rules and probably changing the rule combination algorithm. The effects of doing both these actions is hard for an average user to comprehend. If Alex has set his RCA as 'deny-overrides' and he wants to add his new rules to permit access during that particular week, he will need to either change the RCA to 'permit-overrides' or change each of the deny rules in the policy. Doing either is not desirable because his deny rules will be bypassed. In the proposed system, Alex can add a policy to the policy set defining his access policies and change the PCA to 'permit-overrides' for the specified period. Doing so will still keep all of Alex's deny rules unmodified and his policy set will allow access when at least one of his policies allow access, which is what he intended to do. This is hard to do in current systems, because PCA cannot be changed according to dynamic requirements. The resulting policy set is also more modular and analyzing such a policy set is easier. Finally, it saves the effort and complexity of analyzing the effects of changing the RCA or policy rules, not to mention restoring the original state once the specified time has passed.

An additional benefit of our framework is that SHC can create index rules using attributes like 'username'[5], 'datatype', and 'data source' to create index rules to quickly select relevant policies when a physician tries to access Alex's PHR. These relevant policies form the TAPS for this access request. Suppose policy P880 contains Alex's disclosure policies, P130 contains data source's policy, P110 contains HIPPA policy, P112 contains the electronic privacy act[6], and P21 contains the SHC's disclosure policies. SHC's index rules for Alex's PHR are shown below :

$< \{`username = Alex'\} : P880 >$

[5]The system can use any pseudonym to link Alex's PHR to his policies.

[6]The assumption here is that the rules in these acts can be encoded in a high level language like EPAL or XACML.

$< \{`datasourceId = 814820'\} : P130 >$

$< \{`datatype = PHR'\} : P110, P112 >$

$< \{*\} : P21 >$

Note that, in the last index rule, the attribute value is left blank, which results in P21 being included every time. Using this efficient evaluation of TAPS, SHC can quickly determine the policies that need to be evaluated for an access request to Alex's PHR. We report some performance results of the efficiency of TAPS evaluation in Section 5.6.

5.5 Prototype Implementation

In this section, we describe the prototype implementation of the proposed framework. The prototype implementation of the framework extends the functionality of the policy language. The implementation is done using Sun's open-source XACML engine implementation, where we implemented additional modules and PCAs using Java. The generated policies are written in XACML. We use the Sun XACML PDP implementation because its loading and evaluation times are both reasonable when compared to other popular XACML implementations like XACMLLight and XACML Enterprise. Its overall performance is much better than XACMLLight and close to XACML Enterprise. A detailed comparison of the three implementations is done in [84].

The authorization policy consists of multiple policy sets. These sets consist of the system policy, the patient policy, and the data source policy. The system can be extended to consider the data accessor's policy to ensure that the obligations associated with the access request will be honored. The authorization module is set up as shown in Figure 10. The 'Policy Load and Evaluation' and 'Ancillary' modules are part of the standard XACML engine and the 'PSS' and 'PCA Selector' (explained later in this section) are added to the XACML engine. To make the proposed model

closely compliant with the existing XACML engine, we have modeled the two new sub-modules as XACML policy sets, so that the XACML policy engine can be used to do these evaluations as well.

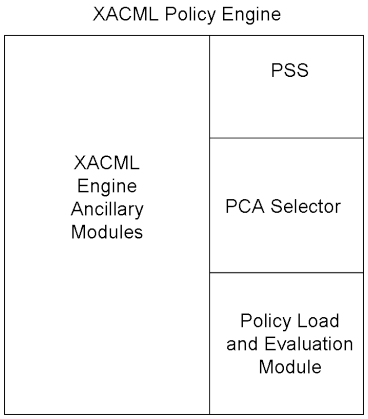

Figure 10: Modified XACML policy engine.

Policy Set Selector (PSS) - The PSS takes the authorization policy as the input, which contains all the available policy sets. The schema of the TAPS as a policy file is shown in Figure 11. It is organized in the Subject, Resource, Action and Environment structure. The PSS evaluates each policy set to find out all the sets that are applicable to this access request. The PCA used here is 'all-that-apply', which is especially developed for the PSS. The function of this PCA is to evaluate all the policy sets and output all that apply. All the policy sets selected by the PSS are stored in a data structure and only those policy sets are considered in the evaluation phase. As mentioned earlier, this reduces the number of policies to be evaluated for an access request and results in considerable run time performance improvement. A detailed discussion of the performance improvement is given in Section 5.6.

PCA Selector - The PCA selector reads the PCA selection file, which is described as a XACML policy. This description is created by the entity that is responsible for making sure that all the relevant policies are taken into consideration. This entity should make sure that the all the available PCAs are encoded as individual policies as shown in Figure 12. This system can be used as a static system by defining the

<PolicySet Combination Algo: all-that-apply>
<Policy Description : Policy 1>
<Subjects>
Required Attribute Sets
</ Subjects >
<Resources>
Policy set or policy
</Resources>
<Actions>
Include Policy set / policy
</Actions>
</ Policy>
<Policy Description : Policy 2>
.....
.....
</Policy>
<Policy Description : Policy 3>
.....
.....
</Policy>
</ PolicySet>

Figure 11: Policy set selector module as a XACML policy set.

selected PCA with no attributes (hence always applicable) and defining all the other PCAs with attributes that are never true. Although such a configuration may not provide some of the key benefits of the proposed framework, it may sometimes be required for backward compatibility.

The PCA selector file is a policy set as shown in Figure 12. All the PCAs are described as contained policy sets and the combination algorithm used is 'only-one-applicable', which is a standard XACML PCA. It returns 'permit' if one of the policy sets is applicable and 'deny' if zero or more than one policy set are applicable. In case the result is 'permit', the applicable policy set returns the name of the PCA to be used in combining policies. This module provides the novel functionality of selecting the PCA dynamically as described in Section 5.3.2.

To continue with the example in Section 5.3, the PCA selection policy set will be set as shown in Figure 12. Initially, when there is no emergency, the PCA 'deny-overrides' will be selected. This will be indicated by the attribute 'emergency' being set to false. When there is an emergency, the attribute is set to true and the PCA

<PolicySet Combination Algo: only-one-applicable>
<Policy Description : Policy 1>
<Subjects>
Required Attribute Sets
</ Subjects >
<Resources>
PCA Algorithm 1
</Resources>
<Actions>
Use this PCA algorithm
</Actions>
</ Policy>
<Policy Description : Policy 2>
.....
</Policy>
<Policy Description : Policy 3>
.....
</Policy>
</ PolicySet>

Figure 12: PCA selector module as a XACML policy set.

evaluation will give the output as 'permit-overrides'. The output PCA again becomes 'deny-overrides' once the emergency is over and the corresponding attribute is set to false.

This attribute can be provided by a number of entities like the 'emergency operations center', the '911 operations center', the patient himself or any other entity that the patient's agent trusts to provide this attribute. Although it sometimes might be difficult to ascertain that this particular patient is involved in an emergency, the patient would give more priority to making his PHR available to medical personnel in an emergency rather than to his privacy. Since the entire system can be audited, any breach of privacy can be discovered on audit.

5.6 Performance Evaluation

In this section, we will discuss the performance evaluation of the various components of the proposed framework. We are basically measuring the following parameters: 1) overhead in evaluating the total applicable policy set (TAPS), 2) overhead in dynamic

selection of the PCA, and 3) time saved in evaluating just the TAPS (and evaluating applicable policies) compared to performing a target match on all the available policies (and evaluating applicable policies).

To measure these parameters, we evaluate the following - 1) TAPS evaluation time vs. total number of available policies , 2) PCA evaluation time vs. number of attributes in each index rule, 3) evaluation time vs. number of policies (with and without TAPS). Reasons for choosing these parameters and the evaluation results are discussed in detail in Section 5.6.2.

5.6.1 Evaluation Setup

In the evaluation setup, we create XACML policies for the modules described in Section 5.5. For evaluating the TAPS, we use the schema shown in Figure 11. We setup a XACML policy file with one index rule representing each available policy file (or policy set). Each index rule contains two attributes, both of which are required for access. There are 16 attributes in total and we select 2 out of them randomly. For the experiments, we use 1,2,4 and 8 index rules for each policy file in each run of the experiment. We also vary the total number of available policies from 1 to 10,000 increasing the number of policies by an order of magnitude each time. Most of the real world policies use 10-20 user attributes coming from the organizations LDAP server [69], [8], hence we feel 16 is a representative number. Moreover, this is a configuration parameter and not a limitation because it can be scaled easily. We also scale the number of attributes in one of the experiments (as described in this Section 5.6.2.2). We believe that most of the real world systems use much less than 10,000 policies. We evaluate performance up to 10,000 policies to observe the system performance over a broad range.

For selecting the PCA, we use the schema shown in Figure 12. Since we have a fixed number of PCA's in the system, we use this evaluation to scale up the number

of attributes from 2 to 10,000 in each index rule. This evaluation gives us an estimate of the evaluation time in a system with large number of attributes.

For evaluating the actual policies, we have created policies with 1,2,4 and 8 rules per policy to be used in different runs of the experiment. We created sets of 10, 100, 1,000, and 10,000 policies.

All experiments were run on a single 2.4GHz Intel Dual Core Pentium machine with 2 GB of physical memory.

5.6.2 Evaluation Results

In this subsection, we present the performance results for the different cases just described.

5.6.2.1 Case 1

In this case, we evaluate the time consumed in evaluating the TAPS with varying number of total available policies. The RCA used is 'all-that-apply', so the evaluation considers all the policies that apply to a particular access request. We change the number of policies from 1 to 10,000 by increasing the number of policies by an order of magnitude in each step. We also vary the number of index rules applicable to each policy to 1,2,4, and 8 in different runs of the experiment. The result is shown in Figure 13. We observe that the evaluations take almost linear time as shown in this semi-log graph. The evaluation time is within 2 seconds even with 1,000 policies with 8 rules each, whereas with 100 policies with 8 rules each the evaluation time is within 250 milli-seconds.

5.6.2.2 Case 2

In this case, we evaluate the applicable PCA from a list of PCAs supported by the system. In our prototype system, we have seven PCAs, each denoted as a policy set with its own index rule. We increase the number of attributes used in each index

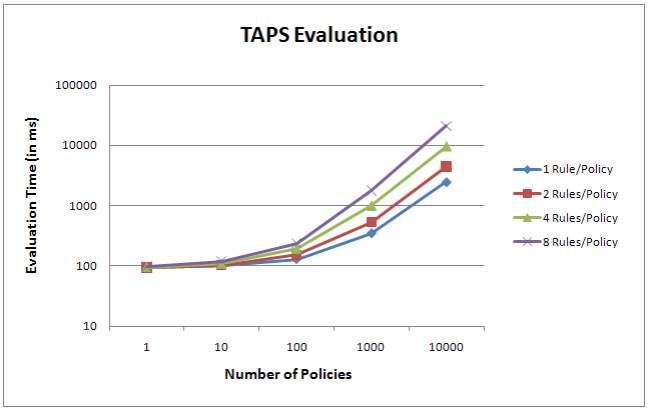

Figure 13: Evaluation time vs. number of available policies.

rule to understand the effect of scaling the attributes on performance. We increase the number of attributes from 2 to 10,000. The run time performance is shown in Figure 14. We observe that even with 100 attributes per index rule, the total evaluation time is under 280 milli-seconds.

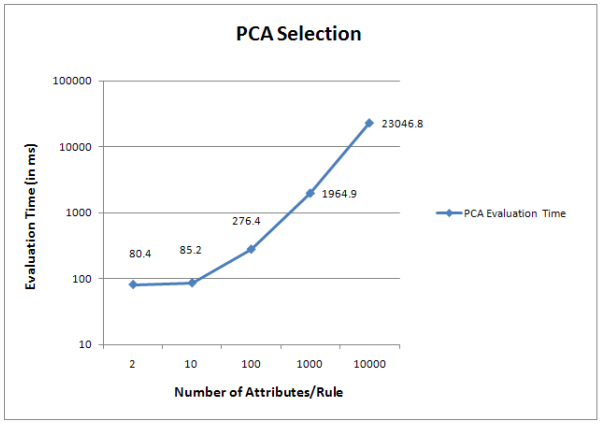

Figure 14: Evaluation vs. number of attributes per index rule.

5.6.2.3 Case 3

In this case, we evaluate the same set of policies with and without the PSS module and compare the performance of the two systems. The setup is described in Section 5.6.1. In each policy file, we have a policy target set up, which is the default method XACML uses to check whether the current policy (file) is applicable to the current request. This target can be set up by resources, subjects, actions, or environments. We set

up these targets with applicable subjects values. This allows us to make a direct comparison with our experimental setup. Also, this does not limit the use of target in the experiments conceptually or physically[7]. We first run the test with all the files and let XACML engine perform target matches with all the available policies and evaluate policies where the target matches. Figure 15 shows the result of this evaluation with about 1% of the policies being evaluated.

For comparison with our proposed system, we run the experiment with the same policy set with the PSS module included. We evaluate the TAPS using the index rule method for all the available policies and force the TAPS to be 1% of the total available policies. The resulting TAPS is stored in an array and the XACML engine then performs evaluation of all the files in this array. The combined time for determining the TAPS and evaluating it is shown in Figure 16. We include 1 percent of the total policies in the TAPS, which we believe is more than what most access requests would require, especially in systems with large number of policies. We chose this percentage so that we have a view of the worst case system performance and expect that most real systems will have fewer policies to evaluate per access request and the evaluation times will be lower that what is observed in Figure 16.

Comparing the results in Figure 15 and Figure 16, we observe that using TAPS evaluation with the index rules and then evaluating the applicable policies is about 4-8 times faster than the conventional method. This is specially important in large systems with a lot of policies. Considering the worst case scenario (10,000 policies, 8 rules/policy), the conventional evaluation takes about 210 seconds compared to 26 seconds on our system. In a more common scenario (100 policies, 8 rules/policy), the evaluation times are 1.8 seconds and 0.5 seconds respectively. We argue that this

[7]Using target in the policy file is optional in XACML. If no target is used, the only way to check the applicability of the policy is to evaluate it and see if it applies to the current request. This will be slower than matching the target and hence we believe that our comparison is fair because we compare our results with the faster version.

performance improvement is not only significant, but critical for real time systems.

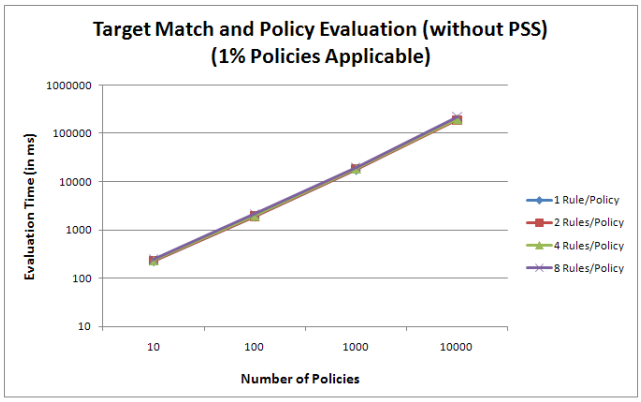

Figure 15: Evaluation time vs. number of total available policies (conventional XACML).

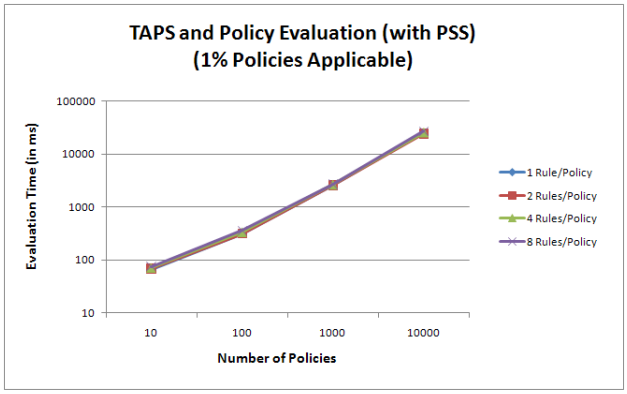

Figure 16: Evaluation time vs. number of total available policies (our proposed framework).

5.6.2.4 Case 4

In this case, we fix the total number of available policies to 1000 and change the percentage of applicable policies to each access request. We perform this experiment with 15access request. We repeat this experiment for 1,2,4 and 8 rules per policy with and without the PSS system and compare their performance. The results are shown in Figure 17 and Figure 18. We observe that in our proposed model the system evaluation time starts from a very low value and increases linearly. On the other hand in existing systems, it starts at near maximum value and remains almost constant.

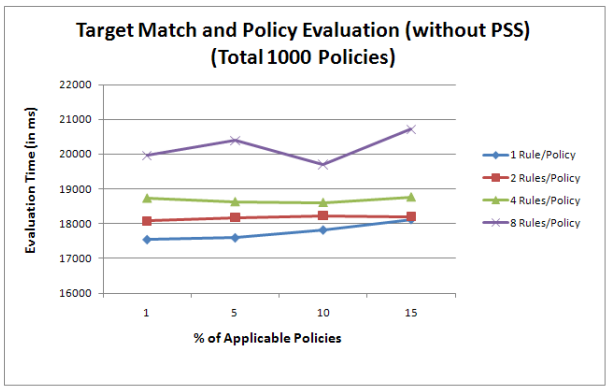

Figure 17: Evaluation time vs. number of total available policies (conventional XACML).

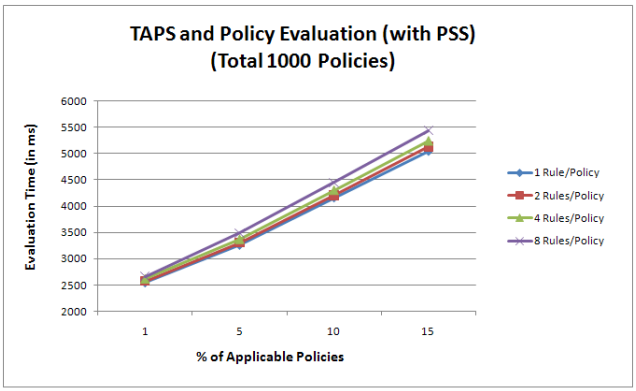

Figure 18: Evaluation time vs. number of total available policies (our proposed framework).

5.7 Related Work

In this section, we review related work in the area of conflict detection, avoidance and resolution works and compare them to our proposed framework.

Mazzoleni, et al., presented a system for integrating authorization policies for different partners organizations [66]. Their core idea is to find the similarity between a set of policies and to use that information to transform the set of policies into a single transformed policy which applies to the request. In their case, the PCA are static there is no way to choose policies dynamically, whereas in our framework we can choose the PCA dynamically. Our framework also allows multiple policies for the same resource, one of which can be chosen at run time.

Another idea for policy conflict resolution in active databases was proposed by Chomicki et al., in [40]. Their system is based on the Event-condition-action paradigm in which policies are formulated using ECA rules. A policy generates a conflict when its output contains a set of actions that the policy administrator has specified cannot occur together. This work is specific to dynamically resolving conflicts among actions in a system, whereas our focus is more on a generic policy-based system to protect the resources. In our framework, the policy composers need not have any idea of the possible conflicts in the system, whereas in Chomicki the system administrator specifically defines conflicting actions. Moreover, in our system there can be a number of authorities who can compose the policies and it is not possible for any one authority to have an idea of all the possible conflicts in advance.

One approach to avoid conflicts in authorization rules is presented by Yu et al., in [90]. They argue that a large number of rules may apply to a service and detecting and resolving conflicts in real time can be a daunting task. Their system is completely static and assumes that is it always possible to determine priorities ahead of time and avoid conflicts. We argue that this is not possible in dynamic environments and is based on multiple factors like the context of the access request, authorities defining the policies, mandatory policies (like regulatory) vs. optional policies, and environmental factors.

Another approach for avoiding conflicts in policy specification is proposed by Agrawal, et al., for defining authorization policies for hippocratic databases [23] and [20]. Their system allows system administrators to specify system policies for administration and regulatory compliance and these policies have the highest priority. Users are allowed to specify their privacy preference as long as their policies do not conflict with the system policies. In our framework, the users can specify their preferences even if they have conflicts with the other policies. The users policies may override other polices or be overridden based on context information. Agrawal's

framework also does not consider changing system and regulatory policies that may create more conflicts with accepted user policies. Also, it may result in removal of conflicts between the new system policy and previously rejected user policies, which is not handled in this system. In our framework, this will be naturally handled without any action on anyone's part to resolve the conflict.

Bertino, et al., presented an approach which is a hybrid of conflict avoidance and conflict resolution [32]. In this work, the authors propose a scheme for supporting multiple access control policies in database systems. Here policies may have 'strong' authorization which are without conflicts or 'weak' authorization with possible conflicts. Compared to this framework, we believe that our approach is more generic because it allows conflicting policies to be composed and resolves conflicts based based on context information. To implement Bertino's proposed system, there should be some static hierarchy (or first specified rule overrides others) for conflict avoidance among strong authorizations. In contrast, our framework will allow dynamic overriding among the authorities.

Another approach to resolving policy conflicts in a hybrid manner is proposed by Jin, et al. [51]. In their work they mention that although resolving conflicts using the static method is easier, it may not be feasible in large systems with large number of policies. The main difference with our framework is that the combination algorithms in their model are defined statically, whereas in our case we decide the combination algorithm at run time based on context information. Also, our framework enables the user to add (remove) PCAs or policies dynamically, an aspect not considered in [51].

5.8 'all-that-apply' Combination Algorithm

Definitions:

$P_i = i^{th}$ Authorization policy.

FID = File Identifier.

FID(P_i) = File Identifier for i^{th} authorization policy file.

TAPS = An array to store FIDs. M-Policy = A policy file with index rules to define applicability of authorization policies.

Algorithm:

```
1   Load M-Policy, Access Request (AR)
2   Define TAPS, initialize i=0, counter=0
3   While (M-Policy (index rules))
4           decision = evaluate(index rule i) against AR
5           if (decision == permit)
6           {       { TAPS[counter++] = FID(Pi) }
7           else
8           {       { continue }
9           increment i
10  return TAPS
```

CHAPTER VI

APPLICATIONS

6.1 Attribute-based rich presence information disclosure system

The attribute-based rich presence information disclosure system is a prototype implementation of a rich presence system supported by an attribute-based authorization system. The rich presence information in this system is composed of a user's real time location information. The prototype is developed leveraging a number of Georgia Tech's services and is designed to work on Georgia Tech's campus only. The services leveraged by this prototype included Georgia Tech's central authorization system for authentication, online directory service for verifying user attributes, and a wireless network based location service called 'whereami'. In this section, we will first introduce the underlying services briefly and then describe the prototype service. A video demonstration of the prototype system is available at [15]. Some applications where this presence information itself can be used to aid access control or provide rich applications are presented in [74], [35].

6.1.1 Underlying services

The prototype system uses Georgia Tech's Central Authentication System (CAS). The CAS is based on the Kerberos protocol that provides a central authentication service for all Georgia Tech services. Online services which are not supported by the institution can still leverage this service by redirecting its users to the CAS portal. The CAS sends a success (failure) message to the service upon successful (unsuccessful) authentication. Each Georgia Tech member has a unique id in the system known as the prism id. This id and its associated password are used to authenticate and

register a user for the prototype service.

The second underlying service leveraged in the prototype is the Georgia Tech directory service. It contains various pieces of information about each user (identified by their prism id) like their name, department, contact number, position in a department, address etc. Out of these, the user's department and position are provided by the system administrator and are verified by him. Hence, these two attributes are trustworthy and are used as verified attributes in our system. The authorization policies are composed using only these two verified attributes.

The third underlying Georgia Tech service leveraged by our prototype system is called 'whereami'. It is a research service provided by Georgia Tech's research network operations center. It provides the location of a user's computer or mobile device connected to the campus Wi-Fi service based on the access point to which the user is currently associated. 'whereami' maintains a list of the MAC addresses of all the devices that are connected to the Wi-Fi service in a central server along with the ids of the access points to which the devices are associated. Users can query the location of their device by sending a query using whereami's programmatic interface APIs. The server returns the querying device's location including their building name, building number and room number with an accuracy of fifty meters.

6.1.2 Prototype system

The conceptual architecture of the system is shown in Figure 19. The main components of this system are the Identity Agent (IdA), the presentity, the querier, location service, databases containing verified attributes, and authentication services for each domain. The presentity is a user who is sharing his presence information and the querier is a user who is requesting the presentity's location information. The software used for this prototype system is symmetric, hence a user can function as a

presentity and a querier simultaneously. However, for the sake of clarity we will describe the system where one user (user 1 in Figure 19) will act as a presentity and another user (user 2 in Figure 19) will act as a querier.

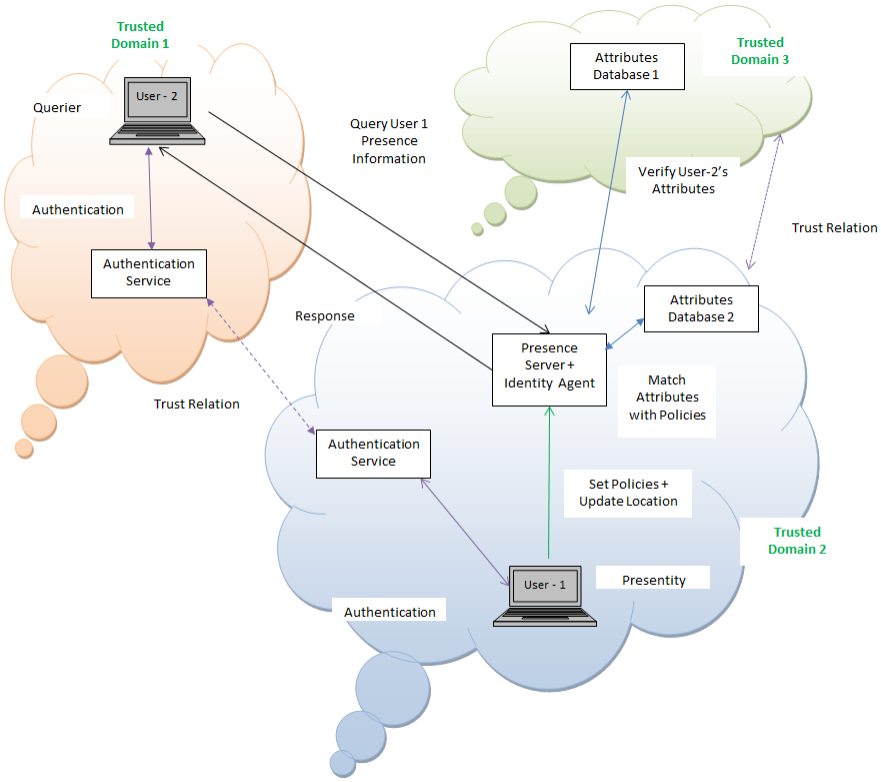

Figure 19: Conceptual architecture of the rich presence information disclosure system.

The prototype system architecture is shown in figure 20. In our prototype system, the location service used is the 'whereami' service, the database of verified attributes is the GT directory service, and the authentication service is the GT CAS. Please note here that user 1 is acting as the presentity and user 2 is the querier. Both the users are in the same domain. A typical flow in the system is as follows. The presentity connects to the IdA, who forwards him to the GT CAS, where the presentity authenticates himself using his GT prism ID and password. Upon successful authentication, the CAS redirects the user back to the IdA and sends the logged-in username to the IdA enclosed in a Kerberos ticket. The presentity then sets his presence information

disclosure policies at the IdA. These policies are maintained by the IdA for each logged user. Thereafter, the presentity queries the 'whereami' periodically for its own location and updates this information in the IdA. The 'whereami' supports only first party lookups, meaning any user can only query his own location. The IdA stores this information for each user. A querier logs in the system using his prism ID and password and wishes to query presentity location. He sends a request to the IdA. The IdA fetches the querier's verified attributes (position and department) from the GT directory and matches these attributes with the presentity disclosure policies. The system supports different granularities of location information and the system can return different granularity of location information based on the level of match between the presentity disclosure policies and the querier attributes. If the querier has all the attributes with the desired values as defined in the presentity disclosure policy, the IdA release the presence information with the highest granularity. If the querier only holds a subset of the desired attributes, the IdA releases a lower granularity of the presence information in accordance with the policy. If the querier holds none of the desired attributes, the location request is denied.

The prototype system has a simple user interface in the client software to authenticate, set disclosure policies and query presentity location. The screen used by the presentity for setting the disclosure policies is shown in Figure 21. It is basically a series of check boxes to select the department and position for which the querier can see the highest granularity of presentity location. The main aim of this interface is to make policy setting very easy, trading off the ability to set more advanced and flexible policies. In this simple GUI screen, the user can select up to four departments and positions to share his location with. For example, if a presentity selects department = 'ECE' and 'CS' and position = 'Professor' and 'Graduate Students', then all the 'Professors' and 'Graduate students' in the 'ECE' and 'CS' department will be able to see the highest granularity of this presentity's location information. If anybody in the

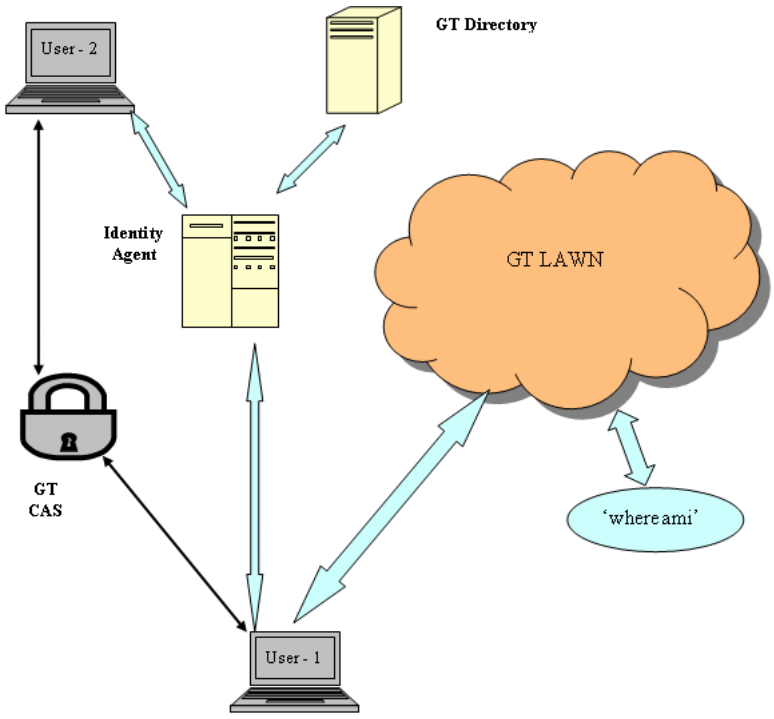

Figure 20: Prototype architecture of the rich presence information disclosure system.

'ECE' and 'CS' department other than a 'Professor' or a 'Graduate student' queries the presentity location, they will receive a lower granularity of location information viz. 'On campus' or 'Off campus'. If a querier from some other department queries the presentity location, then his location request will be denied.

6.2 MedVault - Emergency responders access to sensitive health information

The storage of health records in electronic format and the wide-spread sharing of these records among different health care providers have enormous potential benefits to the U.S. healthcare system. These benefits include both improving the quality of health care delivered to patients and reducing the costs of delivering that care. However, maintaining the security of electronic health record systems and the privacy of the information they contain is paramount to ensure that patients have confidence in the use of such systems. MedVault is a framework for electronic health record sharing that is patient-centric, i.e. it provides patients with substantial control over

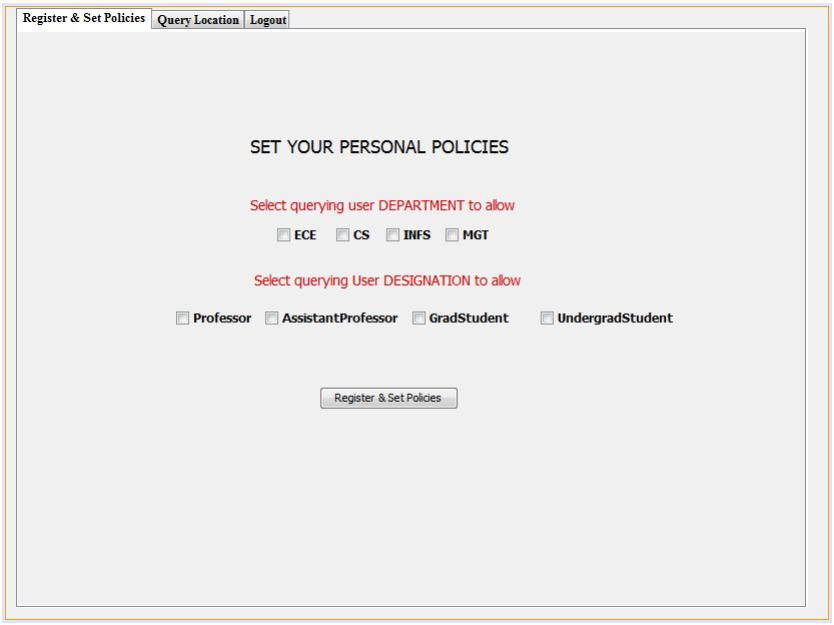

Figure 21: GUI screen for setting the presentity information disclosure policies.

how their information is shared and with whom; provides for verifiability of original sources of health information and the integrity of the data; and permits fine-grained decisions about when data can be shared based on the use of attribute-based techniques for authorization and access control. MedVault framework is a platform for security and privacy research and a number of research ideas have been implemented by different researchers on this platform. The research conducted as part of this dissertation is related to authorization and access control for this framework. MedVault's authorization system is an attribute-based system developed using XACML. In this section, we will briefly introduce the MedVault framework, its authorization system and describe its use within a scenario involving emergency responders' access to health record information.

In this section, we will only cover MedVault's authorization system in detail and will cover the other modules very briefly for understanding purposes only. For a complete description of the entire prototype system and background research publications please refer [67].

6.2.1 System architecture and concepts

The system architecture of the MedVault sharing framework is shown in Figure 22. The health records reside in a source verifiable repository. Suppose a user wishes to access the records of a particular patient. The requesting user connects to his user agent using some generic interface. In our prototype system this is a web browser. The user agent makes an access request to the patient agent on the user's behalf. The Health Information Service allows the querier to locate the available repositories and the types of records available for the patient in question. Upon receiving a request from a user agent, the patient agent notifies the user agent of the attributes required to complete the access. The user agent then aggregates the relevant attributes from a set of Attribute Providers (APs). APs put these attribute values into signed digital credentials. The APs' public keys can be presented to the verifying authority, (in this case the patient agent) in a certificate signed by a Certificate Authority (CA), or they can be gathered a priori through other channels when the patient agent has an existing trust relationship with an AP. The patient agent, authorization module, and access policies are co-located with the repository, either logically or physically depending on the implementation. The patient agent mediates access to the repository and enforces the patient's authorization policies. For an approved access, it also sends the health record information back to the user agent, which relays it back to the user's local device.

Source verifiability and integrity of health records - With the increased reliance on electronic health records for health care delivery, it is necessary that the health care provider who is providing treatment be able to verify the authenticity and integrity of these records. It is in the best interest of the health care provider and the patient that these records are accurate and verifiable. Bauer et al. proposed a method to achieve offline source verifiability and selective disclosure of credentials in [27]. This proposed method was implemented in the MedVault framework to provide a means

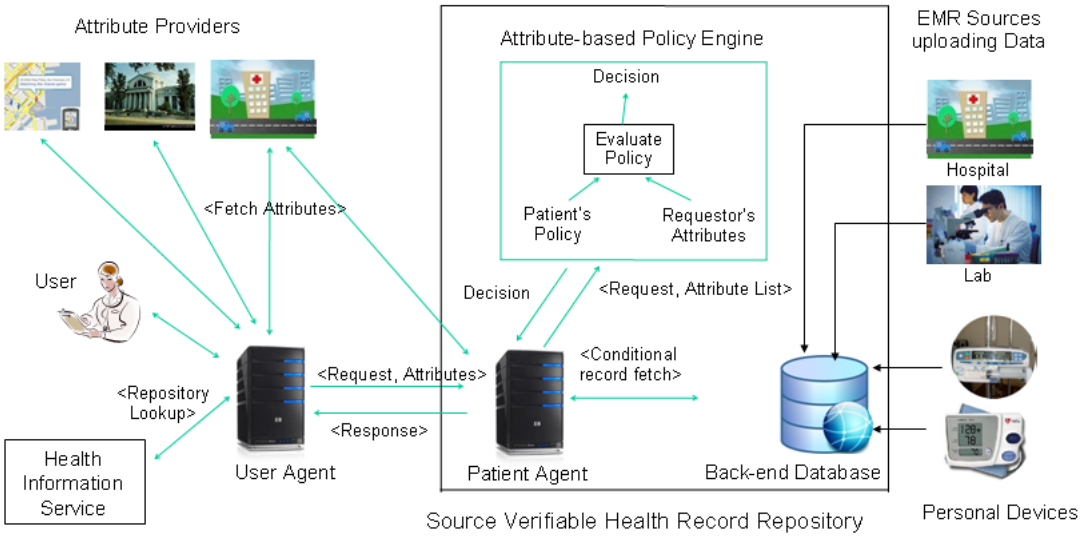

Figure 22: Architecture of MedVault sharing framework.

of source verifiability and integrity check of health records.

Attribute-based authorization system **-** In attribute-based systems, a user has to prove that he holds a set of desired attributes (with specified values) to be authorized to access the desired resources. A unique feature of our approach is that we combine the use of quasi-static attributes like the role of the user or the name of the user's employer with highly dynamic attributes such as the user's location, the time, and characteristics associated with an emergency situation (see Section 6.2.3 for an example of this). Attribute providers (AP) are entities that verify users' attributes and certify them. They create digitally signed credentials and provide them to the user. Our definition of an AP differs from that of an identity provider (IdP) in that an IdP only certifies identity related attributes and a user usually has a small number of IdPs. On the other hand, there could be many APs providing attributes about a given user and these attributes may or may not be identity related. A host of possibilities for APs exists. APs could operate under direct control of the user, functioning similarly to an IdP, or they could be aggregators and providers of publicly-available information, or they could be business or government entities exchanging information

under contractual agreements. Other viable models for APs will undoubtedly arise, as well. There are multiple modes in which attribute information can be gathered from APs and supplied to the patient agent. In one mode, the user agent can retrieve the (digitally signed) attributes and present them to the patient agent. In a second mode, the patient agent can be responsible for collecting attributes. This can be done under explicit authorization from the user via a cryptographic token given by the user agent to the patient agent and forwarded to an AP. Alternatively, the patient agent might retrieve attributes from APs that hold publicly-available information about the user, or it might contact APs with which it has contractual agreements and thus not require explicit authorization from the user. A final mode of operation uses a combination of user-agent-supplied and patient-agent-retrieved attributes. For example, the user agent might present static attributes in the form of a digital credential and the patient agent might be responsible for querying dynamic attributes. An AP can belong to a broad range of entities. For example, an AP could be a medical licensing board that can certify the role of medical professionals like doctors, EMTs, and nurses, or a location service that can certify the current location of the user, or an employer certifying its employees' association with the organization. In the case of the licensing board, the attribute has long term validity. It can be pre-fetched and stored in the user agent. However, highly dynamic attributes like location must be fetched in real time.

Basis in health care systems - Although our system is a prototype running on machines in our research laboratory, it is carefully designed with actual health care systems in mind. The database schema used in our prototype health record repository is based on VistA, which is the electronic health record system used by the VA Hospitals. The concept of a patient agent controlling access to a patient's health records is specifically designed to work either with a personal health record repository, such as GoogleHealth [7] or Microsoft HealthVault [9], or with a community health

record repository. Other aspects of the design, as well as the demonstration scenario described in detail in Section 6.2.3, follow key ideas presented in several use case scenarios developed by the American Health Information Community for the U.S. Department of Health and Human Services (HHS) [4].

The primary HHS use cases upon which our design is modeled are 'Consumer Empowerment: Consumer Access to Clinical Information' and 'Emergency Responder - Electronic Health Record'. The concept of an HIS, including a patient/repository directory service, is present in these use cases, as are entities very similar to our attribute providers (although that terminology is not used). Several aspects of our policy-based and patient-controlled information disclosure approach are in close agreement with the 'Consumer Access' use case. Finally, many of the steps in the demonstration scenario described in Section 6.2.3 closely follow aspects of these use cases. Thus, we are confident that both the concepts and design of our prototype system are transferable to the electronic health record sharing environments that are emerging in the U.S. today.

Security model and assumptions - The Medvault sharing framework is designed with a large, high-level infrastructure in mind. While it inherits a great deal of security from its components, (including standard cryptographic functions, minimal-disclosure credentials, and redactable signatures), our focus is generally on the larger architecture. We assume that all cryptography used is secure, that there exists a trusted PKI infrastructure for professional/licensed entities (i.e., hospitals, doctors, and emergency personnel), that trusted entities are not compromised, and that trusted authorities exist to certify employment, position, emergency situations, the location of entities, and other such attributes. Attribute Providers are trusted by the patient agents that consume their attributes. The Health Information Service is globally trusted to carry out its patient lookup service correctly (failure to do so could lead to denial of service but no leakage of sensitive health information). Agents

are trusted by whomever they are acting on behalf of. The primary privacy threat addressed is the release of either: a) too much information from a patient's records to an authorized individual, or b) any information to an unauthorized individual. A second threat addressed is the modification and/or forgery of records. While untrusted records can have some usefulness, medical personnel may not bother to look up patient records if they are likely to be forged or otherwise unreliable. Providing these two properties, i.e. fine-grained patient control over information disclosure and data verifiability/integrity, allows both patients and health care providers to have enough trust in the system to use it.

6.2.2 Description of prototype system implementation

The prototype system consists of four major modules, viz. the back-end database, identity agents, authorization module, and health information service. In this section, we will describe these four modules and the prototype system interfaces.

Back-end databases - The current prototype system uses a sample database based on the VistA electronic health record and health information system [ref]. The health records are stored in a back-end database, which is a MySQL database instance in our prototype. VistA is built on a client-server architecture, which ties together workstations and personal computers with graphical user interfaces at Veterans Health Administration (VHA) facilities, as well as software developed by local medical facility staff. VistA also includes the links that allow commercial off-the-shelf software and products to be used with existing and future technologies. This provides sufficient motivation to use VistA for modeling our sample database as it can be readily extended for future extensions to the MedVault project.

Identity agents - There are two types of identity agents used in the system: user agents and patient agents. User agents operate on behalf of people trying to access a patient's medical records, including medical personnel, friends, and family. Patient

agents mediate access to said records, and are described in detail below. User agents are not strictly required, but we use them in our prototype system for symmetry. User agents hold credentials, perform patient searches, handle various lookups, and can act as Web proxies for their users. The primary function of a patient agent is to mediate access to the patient's health records.

Authorization module - TheMedVault policy engine is built using XACML, an OASIS standard that uses XML schema for representing authorization and entitlement policies [2]. The schema facilitates inclusion of custom attributes in the policy that are verified while making access decisions. Some terminologies: a resource refers to the object to which access is requested; a subject refers to the user that has requested access to a resource, and an action describes what action a subject wants to take on a particular resource. Attributes are ways to describe a subject/resource/action.

Our current prototype categorizes major parts of a patient's health records into three resources, namely Chronic Conditions, Prescriptions, and Other. A subject could be a doctor or other medical professional (e.g. EMT or nurse) wanting to access a patient's records and an action could be a read or write on the resource. In the current demo, attribute providers provide five attributes, including both quasi-static and dynamic attributes. Two (static) attributes specify the role and the employer of the person requesting access to the resource (e.g. doctor, nurse, EMT). A third (dynamic) attribute indicates where the request is from (location). The last two attributes are related to the emergency response scenario described in Section 6.2.3. The fourth (dynamic) attribute indicates the organization that has been assigned to respond to an incident, e.g. county fire department, ambulance company, etc. The final (dynamic) attribute specifies the location of a patient involved in an incident. (In most cases, this is initially the same as the incident location, but it changes as the patient is transported to an emergency care facility.) There is a default patient policy,

which is used to govern access to patients' records in the absence of explicitly specified preferences. Patients are also provided with an interface to set policies themselves and for the subsets of policies that the patient doesn't (or chooses not to) set, the default policies are applied and the policies are written to an XML policy file, which governs access to the resources. Our current policy specification interface is fairly limited and so we do not describe it herein. We are actively researching the design of a simple interface for policy specification that maintains most of the power and flexibility describable in XACML. In addition to a patient's XML policy file, there is a separate XML policy file that describes the system policies. As an example, suppose there is an accident involving patient X and the policies enable doctors to access his EMR only when they are at the hospital and EMTs to access the record when they are within 1 mile of the accident site. When doctor Y wants to access the EMR of X, she contacts X's patient agent and queries the policies set by X. The patient agent of X lets Y know what attributes are required to access the resources for X (in this case, her role and her location). Y then returns to the agent with these attributes, certified by one or more attribute providers. Next, the policy engine generates a request XML with the subject and attributes. The request is matched against the policies and a permit or deny decision is returned by the policy engine in the form of an XML response.

Health information service - The health information service is a lookup service, which has information about how many health record repositories each patient has, where these are located, and what document categories are contained in these repositories. This information is linked to a unique patient ID for each patient and the HIS can be queried using this patient ID. Although, in the current prototype, we assume that each patient's ID is known to the querier, this may not be the case for a real system. Thus, our future plans include implementing a patient locator service as part of the HIS, which will be able to connect to health registries. These registries

would store patients' information, like SSN, driver's license number, address, etc., and map it to the patient's ID. It should be possible to query these health registries with a parameter like a driver's license number and get the patient's ID in response. This concept of a registry is consistent with the vision spelled out in the use case entitled 'Emergency Responder - Electronic Health Record', available on the Department of Health and Human Services' Web site [4].

6.2.3 Demonstration scenario

The functionality of the system is demonstrated by a real life scenario, where an emergency responder accesses a health record repository to access documents required to provide emergency care at an incident location. In this scenario, there are four actors, a patient involved in an incident, a man passing by the incident location, a 911 operator, and an EMT who is dispatched to the incident location. In this demo, the patient's policy requires two attributes with specific values for access, and the system policy requires three additional attributes. The five attributes are 'user role', 'user location', 'incident location', 'user_RespondedToIncident' and 'user_AffliateEmployer'. Several attribute providers are implemented to certify the values of these attributes for the querier.

The patient's policy for the role 'EMT' states that:

- An EMT can never access documents in category 'Other', and
- An EMT can access documents in categories 'Chronic conditions' and 'Prescriptions' if a valid emergency condition exists, as defined in the associated system policy.

The system policy is defined for critical access permissions. In the demo, the system policy states that:

- an EMT can access 'Chronic conditions' and 'Prescriptions' documents when:

– he is an employee of an affiliate institution,

– that institution is responding to an emergency, and

– he is close enough to the incident site,[1]

As the ambulance is en route to the incident, one of the EMTs logs in to his agent using his username and password. The EMT then enters Carl's name and address to query the HIS about his patient ID and available repositories. The HIS sends back a list of available repositories for Carl. It also includes the categories of documents available in each of them. In the first case, we demonstrate that the EMT who is deputed to the incident tries to access Carl's documents and is able to do so. The system automatically checks the signature of the providing doctor and indicates to the EMT that Carl's health records are authentic. The EMT sees the records, which contain a prescription for insulin. He then views the doctor's notes from Carl's last visit, which indicate that Carl has been having some trouble controlling his blood sugar levels.

In the second case, we demonstrate the situation where another EMT who is not deputed to the incident but is called in at the last minute to assist with the emergency. Initially he tries to access Carl's EMRs when he is far from the 'incident location'. He tries to access records for Carl Johnson by contacting his agent at the repository and sending his required attributes. Since he is far from the 'incident location', the request is denied. Later, when this EMT moves closer to the 'incident location', he retries and this time his access request is granted because the system policy permits this access.

When the EMTs arrive on the scene, they already have a tentative diagnosis of diabetic shock. They immediately test Carl's blood sugar, which indicates that he is hypoglycemic, and they administer glucagon, which reverses the effects of insulin.

[1] 'Close enough' is defined by a configurable parameter, which is set to one mile in our current demonstration setup.

Carl's vital signs begin to improve almost immediately and the EMTs place him in the ambulance and transport him to the ER. During the ambulance ride, Carl regains consciousness but is quite disoriented. He is brought to the ER, where he continues to improve and is given some follow-up treatment. During this time, the ER doctors are able to access Carl's complete health records. When Carl is judged to be stable, he is checked into the hospital for a few days so that his condition can be monitored to try to determine a corrective course of action for his medication. In this example scenario, we see that access to the patient's health records can be critical for early diagnosis. In this specific example, a delay in diagnosing the cause of unconsciousness might actually have been fatal. On the other hand, the carefully defined policies, together with the ability to verify both static and dynamic attributes (the latter in real time), ensures that the patient's privacy is protected by giving access only to EMTs who are dispatched to this emergency, rather than to all EMTs at any time. In addition, the system is auditable, based on log files that maintain details of all access requests.

6.3 An Attribute-based dynamic authorization system for NHIN CONNECT

In this section, we discuss the integration of the dynamic authorization framework with an open-source implementation of the Nationwide Health Information Network (NHIN) specifications. In Section 6.3.1, NHIN specifications and the open source implementation called CONNECT are introduced briefly. In Section 6.3.2, the details of integrations are discussed, and finally in Section 6.3.3, the demonstration scenario is presented.

6.3.1 Introduction to NHIN CONNECT

The US Department of Health and Human Services proposed the NHIN [10] to provide a secure, nationwide, interoperable health information infrastructure that will connect providers, consumers, and healthcare professionals. This will enable health

information to follow the consumer, be available for clinical decision making, and support appropriate use of healthcare information beyond direct patient care so as to improve health. The NHIN specifications describe the different modules of the framework in detail. It is expected that several players will develop products for information exchange in the nationwide health information exchange. NHIN is proposed as the framework to support this exchange. All the vendors who develop NHIN products will have to comply with these specifications. There are four major categories of participants in the NHIN -

- care delivery organizations,
- consumer organizations that operate personal health records,
- health information exchanges, and
- specialized organizations like research organizations or organizations that provide secondary use of data.

The NHIN specifications are located at [11]. There are fourteen different specifications, each focusing on a specific module of the framework. The specifications most closely related to the current research are the NHIN Authorization Framework Specification and NHIN Access Consent Policies Specification.

The NHIN conceptual architecture is shown in Figure 23. It can be viewed as being composed of two elements. First, the zones which define the actions and responsibilities of the NHIN participants, and second the architectural components that exist in these zones and provide services based on the actions and responsibilities of these zones. The NHIE zone contains the geographically distributed participating NHIE systems, which are governed by the local rules and regulations. These NHIEs are interconnected through the NHIN security zone as nodes following a common protocol to exchange health information securely. The main NHIN architectural components contain the NHIN network, the NHIN gateway and the NHIN security zone. The NHIE

network represents a collection of organizations that exchange information within a single NHIE. The NHIN gateway is an implementation of the web-services interfaces for communicating over HTTPS to exchange information securely with other NHIEs. The NHIN security zone creates a virtual secure environment among the gateways. The primary mechanism used to support this is the public key infrastructure.

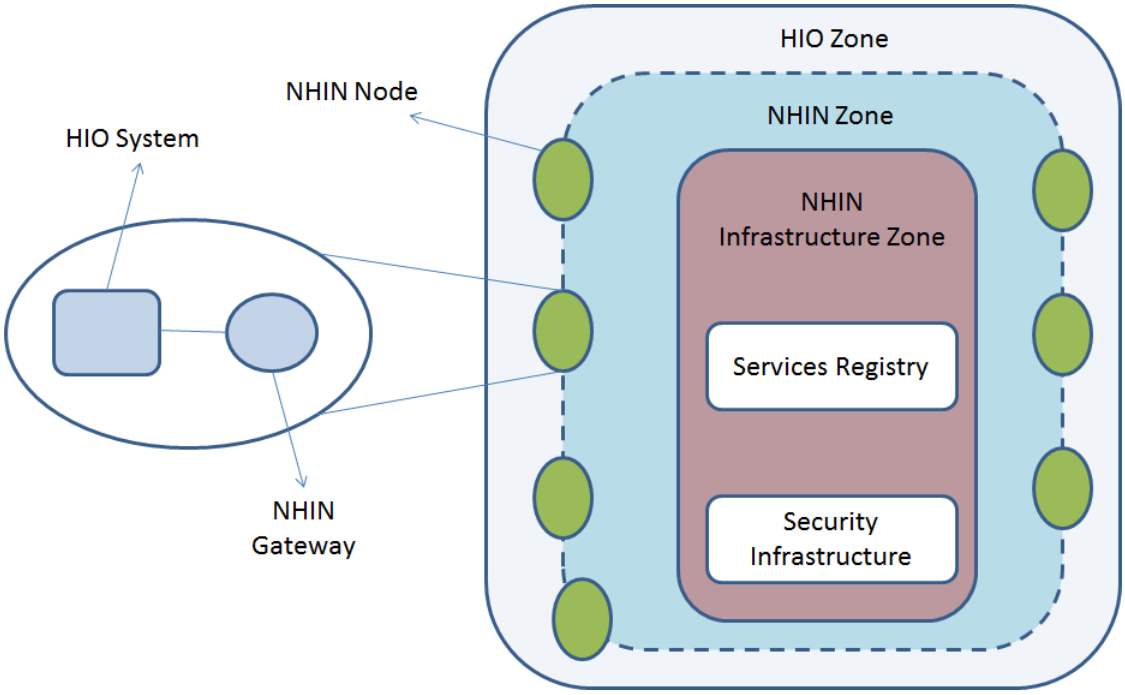

Figure 23: Conceptual architecture of the nationwide health information network.

CONNECT is an open source software solution that supports health information exchange. CONNECT uses NHIN specifications and governance to make sure that data exchanges are compatible with other exchanges being set up throughout the country. This software solution was initially developed by federal agencies for their internal use but is now made available to all organizations to set up their own HIEs. CONNECT is composed of three primary components - core services gateway, enterprise services components, and universal client framework. The Core Services Gateway provides the ability to locate patients at other organizations, request and receive documents associated with these patients, and log these transactions for subsequent

auditing. The NHIN Interface specifications are implemented within the gateway. The Enterprise Service Components provide default implementations of many critical enterprise components required to support HIEs. Products based on CONNECT are free to either use these components or develop their own custom implementations for these modules. The Universal Client Framework contains a set of applications that can be used as a demonstration system for the gateway solution. This provides a convenient platform to innovate on top of the existing CONNECT implementation.

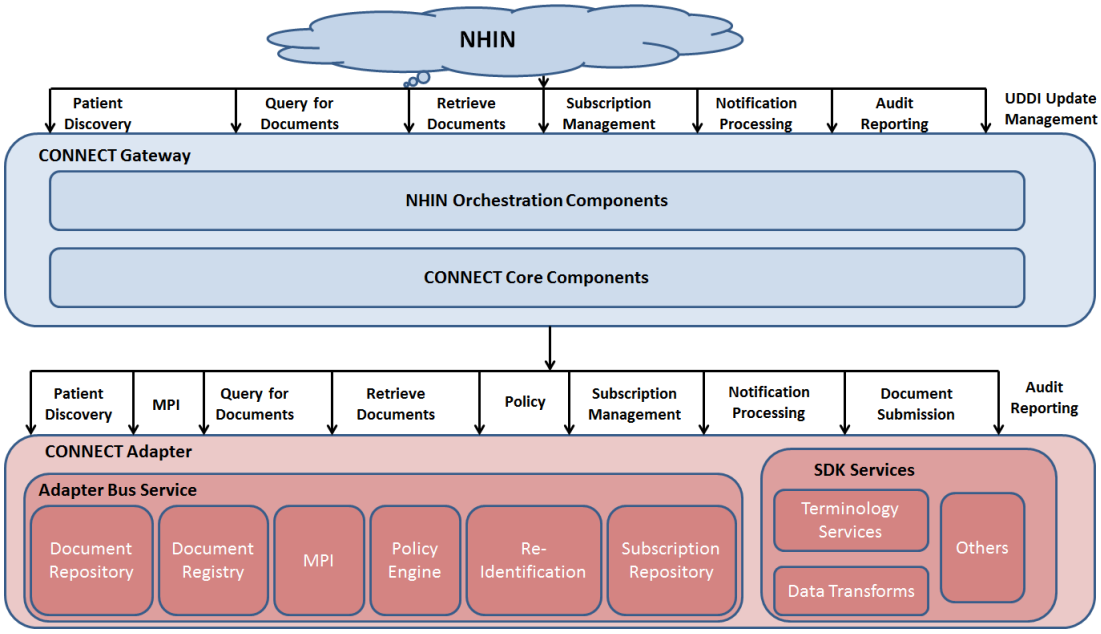

Figure 24: High-level view of the CONNECT architecture for a message received from the NHIN.

6.3.2 Integrating Dynamic Authorization System with NHIN CONNECT

In this section, some of the specific details of the NHIN CONNECT system architecture related to the authorization mechanism are presented. This is followed by the details of how the NHIN CONNECT is deployed and the integration of the dynamic authorization system with NHIN CONNECT. CONNECT v3.0 is used for the integration.

The default authorization mechanism in NHIN CONNECT is based on OpenSSO [14].

OpenSSO provides authentication and authorization services for CONNECT. It also provides a circle of trust among different web-services components of CONNECT for building and maintaining trust among them. This is based on the public key infrastructure. OpenSSO and its associated tools are provided in CONNECT as third party components. These components need to be integrated with the CONNECT deployment. A set of scripts are provided to achieve this integration.

The NHIN policy engine is a large complex entity, where the policy decision point (PDP) and the single sign-on authentication components are based on OpenSSO. The CONNECT policy engine enterprise service component is shown in Figure 25. The remote gateway is the NHIN gateway for the organization requesting the resources. The request from the remote gateway is intercepted by the NHIN orchestration components in the NHIN gateway, which is forwarded to the adapter policy. The adapter policy forwards the request to the policy engine orchestrator, who in turn forwards it to the policy enforcement point (PEP). As the name suggests, the PEP is the entity which is responsible for the enforcement of authorization policies. CONNECT is designed so that either the default OpenSSO PDP component can be used, or any other specific implementation can be chosen by a particular vendor. The particular PDP component being used is described in a configuration file. The PEP reads this configuration file and forwards the request to that particular PDP. In case of the default PDP, the request is passed to the OpenSSO PDP. The OpenSSO PDP can either use a customized implementation, where the policies can be set up using drop down boxes in the web server hosting CONNECT. The other (and more popular) option is to used the XACML based OpenSSO PDP, which uses XACML policies to perform access control. The interface uses a Java class which imports the required packages from OpenSSO and queries the patient consent documents from the repository. The adapter document repository is the one which holds these patient consent

documents. These documents can be created by using the patient consent management GUI (shown in Figure 25). The policy information point (PIP) is used to store (retrieve) these documents from the repository to the GUI. The PIP also interfaces with the PEP to pass these documents to the PDP, which uses this patient consent information to make authorization decisions.

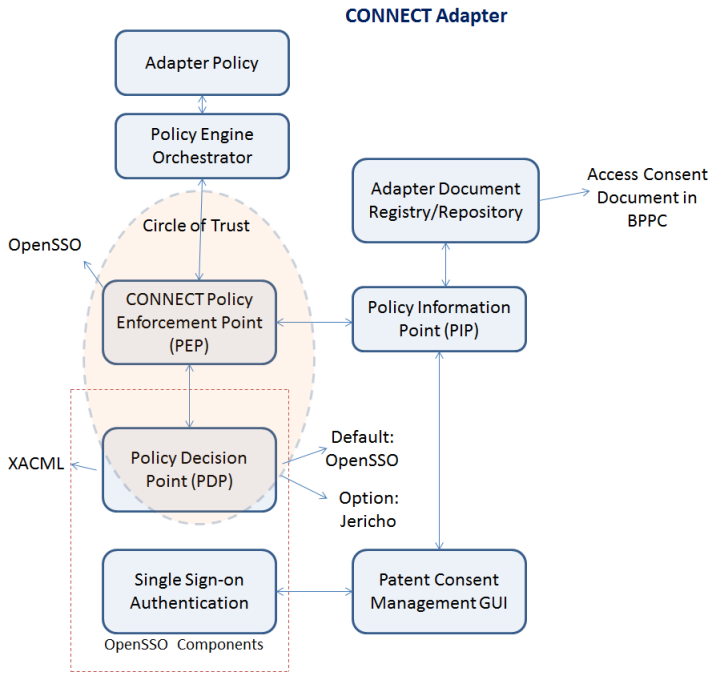

Figure 25: Policy engine enterprise service component.

The CONNECT software is available for download at the CONNECT website [13]. It can be downloaded to a local system and build using the ant environment. The target is then deployed in the GlassFish web server [6]. Restarting the GlassFish web server runs the CONNECT installation. The CONNECT gateway requires a good configuration server with the minimum hardware requirement being a 2GHz processor, 4 GB RAM and a 100 MB Ethernet interface. CONNECT has different versions for Windows and Linux. It requires the Java SDK server platform, GlassFish application server, soapUI server platform and a relational database. Self-signed certificates can be used to run the gateway in test mode, but if the NHIN gateway is to exchange data with other remote gateways, then the NHIN node has to be register with the

HIE and a signed certificate is required for the NHIN node to be operational.

The integration of OpenSSO PDP and the dynamic policy engine is achieved by keeping the OpenSSO PDP as the default policy engine and attaching the dynamic policy engine as an add on module. The integrated system is shown in Figure 26. In v3.0 of NHIN CONNECT, the PDP just considers the 'OptIn' or 'OptOut' status stored in the patient consent document to make authorization decisions. This dynamic authorization system on the other hand considers a number of static and dynamic attributes to reach on an authorization decision. Details of these attributes and an example scenario are presented in detail in Section 6.3.3. The current NHIN gateway has a set of eight attributes that it receives from the remote gateway. On the other hand, the dynamic authorization system can potentially use an unlimited number of attributes for authorization, provided they are verifiable. In Section 6.3.3, we present a scenario where a few common attributes are used for authorization. In the integrated system the OpenSSO PDP combines the consent status in the consent document and the authorization decision of the dynamic authorization system to decide on the final decision. If the consent status is 'OptIn' and the dynamic authorization is 'Permit', then the OpenSSO PDP returns the final decision as 'Permit'. In all other cases, the returned final decision is 'Deny'.

The NHIN CONNECT implementation comes with a soap-based test suite to verify its functionality. We use the same test suite to verify the functionality of the dynamic authorization system as illustrated in Section 6.3.3.

6.3.3 Demonstration Scenario

Setup - The demo setup runs on the lab server. Laptop and desktop computers are used as client machines to ssh into the system and interact with it. The server runs the GlassFish web server with NHIN CONNECT running in the default domain. As depicted in Figure 26, the dynamic policy engine sits at the back end and does not

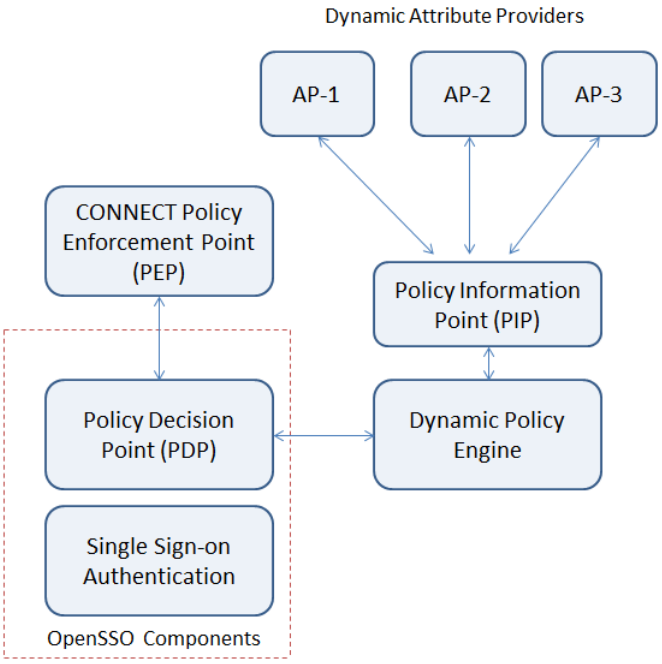

Figure 26: Dynamic policy engine's integration with OpenSSO policy engine.

interact with the front end GUI. The client system runs an Xserver for XForwarding the SoapUI to execute the test cases from the client machine. An instance of this UI is shown in Figure 27. A suite of test cases designed for testing OpenSSO with NHIN CONNECT is developed. We load this test suite into the SoapUI.

Scenario - In the demonstration scenario, we use a sample patient called 'Gallow Younger' with a document in the repository with an identifier 'D123401'. The requester in the scenario is called 'John Smith' which sends his 'community-id' and 'authority-id' with the request for accessing the document 'D123401'. In the demonstration scenario, our aim is to verify the operation of the dynamic policy engine and as such we send these requests to the policy engine which returns its response as a 'Deny' or 'Permit'. In the setup, the OpenSSO policy engine checks the 'opt-in' or 'opt-out' status of the patient and makes a decision based on that. The dynamic policy engine uses two dynamic attributes (as explained in the next paragraph) to reach on its decision. These two decision are combined together to reach on a unique

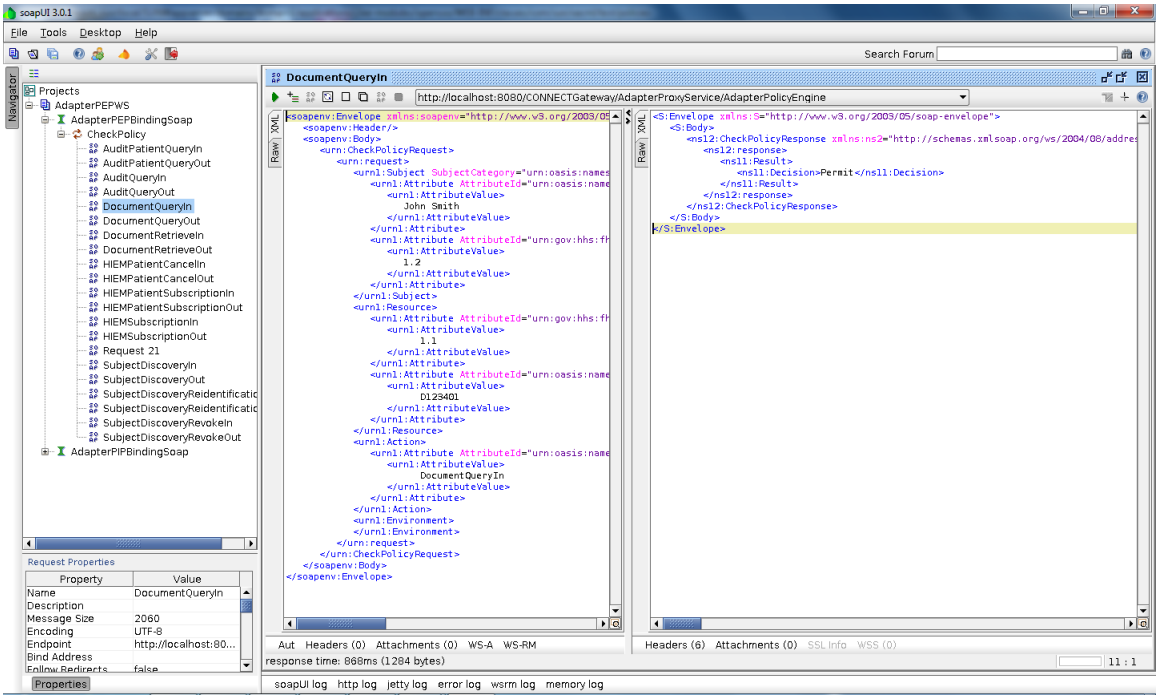

Figure 27: The SoapUI interface used to verify the dynamic policy engine integration with OpenSSO.

decision for the entire policy engine. In this setup, we use a logical 'AND' combination to combine the two decisions, meaning that if both the decisions are 'Permit', then the final decision is 'Permit'. In all other cases, the final decision is 'Deny'.

The demonstration scenario for this setup is as follows: A patient called 'Gallow Younger' signs his 'opt-in' consent for his document with id 'D123401'. He has a policy with the following rules -

1. *'In an emergency, an EMT can access his documents'*,
2. *'A doctor should be able to access his document irrespective of the emergency status'*.

The dynamic authorization engine uses two attributes called 'role' and 'emergency' in its policy. The attribute 'role' takes the values 'Doctor' or 'EMT', while the attribute 'emergency' takes the values 'true' or 'false' to indicate if there is an emergency situation or not.

Results - We run different test cases to verify the behavior of the dynamic policy engine. The patient consent status has been set to 'opt-in' by him, so the final decision will depend on whether the dynamic policy engine sends a 'Permit' or 'Deny'. We then change the patient consent status to 'opt-out' and then irrespective of the decision from the dynamic policy engine, the final result is 'Deny'. The attribute values for the test cases and the final decision of the authorization engine are shown in table 1.

Table 1: Authorization test cases

Case	Consent-status	Role	Emergency	Final-decision
1	opt-in	Doctor	false	**Permit**
2	opt-in	EMT	false	**Deny**
3	opt-in	Doctor	true	**Permit**
4	opt-in	EMT	true	**Permit**
5	opt-out	Doctor	false	**Deny**
6	opt-out	EMT	false	**Deny**
7	opt-out	Doctor	true	**Deny**
8	opt-out	EMT	true	**Deny**

CHAPTER VII

CONCLUSION

Attribute-based authorization is rapidly gaining popularity in the research community as well as in systems developed for real world applications. Real world implementations do not exhibit the full capabilities of attribute-based authorization systems like rich expressiveness, dynamic functionality or use of a large number of authorization attributes for fine grained access control. This is because the current systems and policy languages only provide basic support for attribute-based access control but do not look into the underlying research issues limiting the capabilities of such systems. Some of these limitations are outlined below -

- Scarcity of trusted attributes that can be used for authorization is a big problem. In most organization, LDAP servers are the only trusted attribute suppliers.
- Authorization systems are not dynamic in nature. Although they do use dynamic attributes for authorization to account for the changing environment, they have very little or no provisions to use that for making the authorization system dynamic and adaptive.
- On many occasions the dynamic environment of access and its resultant threats can be addressed more effectively if the different policies governing the access are combined by selecting an appropriate combination algorithm at runtime.
- Determining the set of applicable authorization policies in an efficient manner.
- Demonstrating the benefits of these novel features in real world applications.

In the research presented in this dissertation, we have addressed these issues by conducting research and proposing practical solutions. We have also implemented

these proposed solutions in a number of systems to illustrate their novel functionality and benefits. We realize that these systems have the limitation that they are research prototypes requiring improvements for scalability and handling heavy traffic. Despite these limitations, they still demonstrate the novel functionalities and highlight the potential benefits that can be achieved in real systems.

7.1 Contributions

Contributions to advance the state of the art have been presented in this dissertation, including

- A novel reputation system model to evaluate trust in authorization attributes aggregated from a wide variety of sources. By leveraging transitive trust, this model makes a large number of diversified verifiable attributes available for authorization.

- The reputation system model is attack-resistant to all the known common attacks making it suitable for use in practical systems.

- A novel attribute-based framework for selecting relevant policies efficiently considering the dynamic context of the access request. This framework provides an efficient indexing method for selecting the applicable policies to each access request providing up to 9x advantage in speed of access and evaluation.

- A novel method to dynamically change the algorithms to combine attribute-based authorization policies from multiple sources based on the context of the access request.

- The proposed attribute-based authorization system is implemented in several application areas like electronic medical records, medical databases containing sensitive information, and rich-presence information. This authorization system provides a number of functionalities like fine-grained authorization, data

hierarchy based protection, context aware authorization and multi-authority authorization.

- The proposed dynamic authorization system is integrated with the open source implementation of the Nationwide Health Information Network called CONNECT.

7.2 Future Work

In this research, we have developed a number of ideas and systems. In addition to contributing to the state of the art, it also opens up a number of avenues for conducting further research. Also, there is scope to improve the research prototypes and to perform further bench marking. In the following sections, we describe some of the possibilities of furthering this research.

7.2.1 AttributeTrust

We have done an analytical study of the AttributeTrust model to verify its attack-resistance against commonly known attacks. We have also performed some preliminary computational analysis by simulating the AttributeTrust reputation system model. This simulation was done on Emulab with a small number of nodes. A larger simulation with a large number of nodes and a more thorough computational analysis will help in understanding the performance of the system. Such an analysis will be helpful in understanding the limitations and possible improvements in the current system. During the preliminary analysis of the simulation results, we realized that the node weights used in the AttributeTrust framework need to be normalized using a suitable factor. This can be one of the contributions of a large scale simulation.

7.2.2 Dynamic authorization system

In the dynamic authorization system, we achieve upto 9x advantage in accessing and evaluating applicable policies. One of the major limiting factors on speed in

the current model is the use of XACML policies and standard I/O interfaces for accessing and evaluating policies. This choice was made in a carefully considered tradeoff between performance and portability. We also focused on keeping the system compatible with existing systems, so that the proposed system can be used as an add-on module to current systems. The benefits of these were seen in its integration with NHIN CONNECT. Some researchers have proposed binary or other efficient formats for storing authorization policies and evaluating them [64]. We strongly believe that using the XACML policies and standards-based policy engine is the right approach. As such, we do not recommend converting the entire system to a binary format and limiting its portability. In some cases, where the authorization systems are run on high end servers that are running continuously and speed of decision making is extremely critical, a mixed approach may be useful. Policies can be loaded and converted to binary format internally, which will later speed up evaluation. There will be some overhead in converting the policies to binary, but if the servers are rebooted infrequently, that overhead will be low. Since the policies can be changed dynamically, care will be required in reflecting those changes in real time to the binary format and also to the XACML policies as the cached binary policies will be lost on system restart or crashes.

7.2.3 Rich presence information disclosure system

The rich presence information disclosure system was built as a prototype to demonstrate the use of attribute-based policies in information disclosure and their use in releasing multiple granularities of data. This system is developed in the context of services at Georgia Tech and is designed to be used only on the Georgia Tech campus. Improvements can be made in the prototype system by adding functionalities like receiving accurate location (in terms of latitude and longitude) from GPS enabled devices to cover a wider geographical area. The system can also use external trusted

attribute providers and other authentication service providers, for example, OpenID providers. This will make the prototype system more general purpose and can be used in a number of real settings.

7.2.4 NHIN CONNECT integration

In the current integrated system, authorization attributes used are from the same domain so they are considered trusted. One of the future work action items can be to use external attribute providers, which can provide verifiable attributes to organizations. Examples of these attribute providers include organizations like health care providers, insurance companies, large public sector employers, which are naturally connected to the patients and health care personnel. A Policy Information Point (PIP) can act as an interface to interact with all these attribute providers and provide these attributes to the dynamic authorization system.

We have also done some preliminary research to study how native medical ontologies can be used to define broad categories for access control. The medical ontologies can provide a mapping between the broad categories and the low level data elements. Since the patients view their data as being composed of a number of groups, which can be categorized in a few broad natural categories based on its sensitivity, they expect to define disclosure policies based on these categories, or their sub-categories. On the other hand, the actual low level data elements are in the form of lab tests, prescriptions, medications, orders, consults, reports, patient reported data, etc. As such, applying the high level category-based access control policies to low level data elements becomes a complex task. We propose the use of medical ontologies to map between these high level categories and low level data elements. Researching this approach and implementing a prototype system is one of the proposed future works.

REFERENCES

[1] *Advogato trust metric*. http://www.advogato.org/trust-metric.html (April 2011).

[2] *AuctionBytes survey*. http://www.auctionbytes.com/cab/abu/y203/m01/abu0087/s02 (April 2011).

[3] *Ebay feedback system*. http://pages.ebay.com/help/feedback (April 2011).

[4] *Emergency Responder Electronic Health Record use case*. http://healthit.hhs.gov/ (April 2011).

[5] *eXtensible Access Control Markup Language (XACML)*. http://www.oasis-open.org/ (April 2011).

[6] *GlassFish*. https://glassfish.dev.java.net/ (April 2011).

[7] *Google Health*. http://www.google.com/health/ (April 2011).

[8] *LDAP Authentication Attributes*. http://docs.sun.com/source/817-7647/ldapauth.html#wp19608 (April 2011).

[9] *Microsoft HealthVault*. http://www.healthvault.com (April 2011).

[10] *NHIN*. http://www.hhs.gov/healthit/healthnetwork/background/ (April 2011).

[11] *NHIN Specifications*. http://healthit.hhs.gov/ (April 2011).

[12] "P3P: The Platform for Privacy Preferences," *http://www.w3.org/P3P/ (April 2011)*.

[13] *CONNECT*. http://www.connectopensource.org (April 2011).

[14] *OpenSSO*. http://opensso.org (April 2011).

[15] *Video demonstration of the rich presence information disclosure system*. http://www.ece.gatech.edu/d̃blough/videos/Rich_Presence.avi (April 2011).

[16] "Department of defense : Trusted computer system evaluation criteria," *Department of Defense*, 1983.

[17] "cagrid roadmap 2.0 working draft," *National Cancer Institute*, 2010.

[18] Abadi, M., *Access Control in a Core Calculus of Dependency*. Electronic Notes in Theoretical Computer Science, 2007.

[19] Abadi, M., *Variations in Access Control Logic.* Deontic Logic in Computer Science, 2008.

[20] Agrawal, R., Bird, P., Grandison, T., Kiernan, J., Logan, S., and Rjaibi, W., "Extending relational database systems to automatically enforce privacy policies," in *ICDE*, pp. 1013–1022, April 2005.

[21] Agrawal, R., Kiernan, J., Srikant, R., and Xu, Y., "Hippocratic databases," *Proceedings of the 28th VLDB Conference*, 2002.

[22] Agrawal, R., Kiernan, J., Srikant, R., and Xu, Y., "Order preserving encryption for numeric data," in *SIGMOD, 2004, Paris, France*, 2004.

[23] Agrawal, R., Asonov, D., Bayardo, R., Grandison, T., Johnson, C., and Kiernan, J., "Managing disclosure of private health data with hippocratic databases," *IBM Research White Paper*, Januray 2005.

[24] Ahn, G. and Sandhu, R., "Role-based authorization constraints specification," *ACM Trans. on Information and System Security*, vol. 3, November 2000.

[25] Alex X. Liu, A., Chen, F., Hwang, J., and Xie, T., "Xengine: A fast and scalable xacml policy evaluation engine," *SIGMETRICS08*, 2008.

[26] Barth, A., Mitchell, J., and Rosenstein, J., "Conflict and combination in privacy policy languages," in *Workshop on Privacy in the Electronic Society*, October 2004.

[27] Bauer, D., Blough, D. M., and Cash, D., "Minimal information disclosure with efficiently verifiable credentials," *Fourth ACM Workshop on Digital Identity Management*, 2008.

[28] Bayardo, R. and Agrawal, R., "Data privacy through optimal k-anonymization," in *ICDE '05 Proceedings of the 21st International Conference on Data Engineering*, 2005.

[29] Bell, D. E. and LaPadula, L. J., "Secure computer system: Unified exposition and multics interpretation," *The MITRE Corporation*, 1976.

[30] Bertino, E., Bonatti, P. A., and Ferrari, E., "Trbac: A temporal role-based access control model," *ACM Transactions on Information and System Security*, vol. 4, p. 191223, August 2001.

[31] Bertino, E., Brodie, C., Calo, S. B., Cranor, L. F., Karat, C., Karat, J., Li, N., Lin, D., Lobo, J., Ni, Q., Rao, P. R., and Wang, X., "Analysis of privacy and security policies," *IBM Journal of Research and Development*, vol. 53, 2009.

[32] Bertino, E., Jajodia, S., and Samarati, P., "Supporting multiple access control policies in database systems," *ACM Transactions on Database Systems*, vol. 26, 1996.

[33] Beth, T., Borcherding, M., and Klein, B., "Valuation of trust in open network," *Proceedings of the European Symposium on Research in Computer Security*, pp. 121–130, 1994.

[34] Biba, K. J., "Integrity considerations for secure computer systems," *The MITRE Corporation*, 1977.

[35] Boyer, J. P., Tan, K., and Gunter, C. A., "Privacy sensitive location information systems in smart buildings.," in *Security in Pervasive Computing*, 2006.

[36] Buchegger, S. and Boudec, J. L., "The effect of rumor spreading in reputation systems for mobile adhoc networks," in *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, March 2003.

[37] Byun, J.-W. and Li, N., "Purpose based access control for privacy protection in relational database systems," *The VLDB Journal*, vol. 17, pp. 603–619, 2008.

[38] Chadwick, D. W., "Authorisation using attributes from multiple authorities," *Proceedings of the 15th IEEE International Workshops on Enabling Technologies*, 2006.

[39] Chivers, H., "Personal attributes and privacy," in *Department of Computer Science, Univeristy of York,Heslington, York*.

[40] Chomicki, J., J. Lobo, M., and Naqvi, S., "Conflict resolution using logic programming," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, Januray/February 2003.

[41] Douceur, J. R., "The Sybil attack," *Proceedings for the 1st International Workshop on Peer-to-Peer Systems*, 2002.

[42] Elliott, A. and Knight, S., "Role explosion: Acknowledging the problem," *WORLDCOMP*, 2010.

[43] EPAL. http://www.w3.org/Submission/2003/SUBM-EPAL-20031110/ (April 2011).

[44] Ferraiolo, D. and Kuhn, D., "Role-based access control," *15th National Computer Security Conference*, pp. 554–563, 1992.

[45] Fisler, K., Krishnamurthi, S., Meyerovich, L., and Tschantz, M., "Verification and change impact analysis of access control policies," in *International Conference on Software Engineering*, May 2005.

[46] Guha, R., Kumar, R., Raghavan, P., and Tomkins, A., "Propagation of trust and distrust," *World wide web conference*, May 2004.

[47] Halpern, J. and Weissman, V., "Using first-order logic to reason about policies," in *IEEE Computer Security Foundations Workshop*, 2003.

[48] Hoffman, K., Zage, D., and Nita-Rotaru, C., "A survey of attack and defense techniques for reputation systems," Tech. Rep. CSD TR #07-013, Computer Science Department, Purdue University, 2007.

[49] Jajodia, S., Samarati, P., Sapino, M. L., and Subrahmanian, V. S., "Flexible support for multiple access control policies," *ACM Transactions on Database Systems*, vol. 26, no. 2, 2001.

[50] Jin, J., Ahn, G.-J., Covington, M. J., and Zhang, X., "Toward an access control model for sharing composite electronic health record," in *4th International Conference on Collaborative Computing*, 2008.

[51] Jin, J., Ahn, G.-J., Hu, H., Covington, M. J., and Zhang, X., "Patient-centric authorization framework for sharing electronic health records," in *SACMAT*, 2009.

[52] Kamoda, H., Yamaoka, M., Matsuda, S., Broda, K., and Sloman, M., "Policy conflict analysis using free variable tableaux for access control in web services environments," in *WWW Conference*, 2005.

[53] Kamvar, S. D., Schlosser, M. T., and Garcia-Molina, H., "The eigentrust algorithm for reputation management in p2p networks," *World wide web conference*, May 2003.

[54] Klingenstein, N., "Attribute aggregation and federated identity," *Proceedings of the 2007 International Symposium on Applications and the Internet Workshops*, 2007.

[55] Koshutanski, H. and Massacci, F., "An access control framework for business processes for web services," in *ACM Workshop on XML Security*, October 2003.

[56] Lampson, B. W., "Protection," *Proceedings of the 5th Princeton Conference on Information Sciences and Systems*, 1971.

[57] Levien, R. and Aiken, A., "Attack-resistant trust metrics for public key certification," *Proceedings of the 7th USENIX Security Symposium*, 1998.

[58] Li, N., Wang, Q., Qardaji, W., Bertino, E., Rao, P., Lobo, J., and Lin, D., "Access control policy combination: Theory meets practice," in *SACMAT*, 2009.

[59] Li, N., Wang, Q., Qardaji, W., Bertino, E., Rao, P., Lobo, J., and Lin, D., "Access control policy combining: Theory meets practice," in *ACM SACMAT*, 2009.

[60] Lin, D., Rao, P., Bertino, E., and Lobo, J., "An approach to evaluate policy similarity," in *SACMAT*, 2007.

[61] Linn, J. and Nystrom, M., "Attribute certification: An enabling technology for delegation and role-based controls in distributed environments," *Proceedings of the fourth ACM workshop on RBAC*, pp. 121–130, 1999.

[62] Lupu, E. and Sloman, M., "Conflict analysis for management policies," in *Proceedings of the Vth International Symposium on Integrated Network Management, San-Diego*, 1997.

[63] Lupu, E. and Sloman, M., "Conflicts in policy-based distributed systems management," in *IEEE Transactions on Software Engineering*, pp. 852–869, Nov/Dec 1999.

[64] Marouf, S., Shehab, M., Squicciarini, A., and Sundareswaran, S., "Statistics clustering based framework for efficient xacml policy evaluation," *10th IEEE international conference on Policies for distributed systems and networks*, 2009.

[65] Masayuki, C., Kenji, U., and Yoji, M., "Personal attribute provider: A secure framework for personal attribute exchange on the internet," *Transactions of IPSJ*, vol. 47, no. 3, pp. 676–685, 2006.

[66] Mazzoleni, P., Crispo, B., Sivasubramanian, S., and Bertino, E., "Xacml policy integration algorithms," in *ACM Transactions on Information and System Security (TISSEC)*, pp. 852–869, February 2008.

[67] Mohan, A., Bauer, D., Blough, D., Ahamad, M., Bamba, B., Krishnan, R., Liu, L., Mashima, D., and Palanisamy, B., "A patient-centric, attribute-based, source-verifiable framework for health record sharing," CERCS Tech Report GIT-CERCS-09-11, Georgia Tech, 2009.

[68] Nagaraj, S., "Access control in distributed object systems: Problems with access control lists," in *IEEE WETICE*, 2001.

[69] Ngo, L. and Apon, A., "Using shibboleth for authorization and authentication to the subversion version control repository system," in *IEEE ITNG*, 2007.

[70] Nykanen, T., "Attribute certificates in x.509," *Tik-110.501 Seminar on Network Security*, 2000.

[71] Rao, P., Lin, D., Bertino, E., Li, N., and Lobo, J., "An algebra for fine-grained integration of xacml policies," in *CERIAS Tech Report 2008-21, Purdue University*, 2008.

[72] Reiter, M. K. and Stubblebine, S. G., "Authentication metric analysis and design," *ACM Transactions on Information and System Security*, vol. 2, pp. 138–158, May 1999.

[73] Rouached, M. and Godart, C., "Reasoning about events to specify authorization policies for web services composition," in *IEEE International Conference on Web Services (ICWS)*, September 2007.

[74] Sadeh, N. M., Gandon, F. L., and Kwon, O. B., "Ambient intelligence: The mycampus experience," in *CMU Report CMU-ISRI-05-123*, 2005.

[75] Sahai, A. and Waters, B., "Fuzzy identity-based encryption," *15th National Computer Security Conference*, 2004.

[76] Sandhu, R., Coyne, E., Feinstein, H., and Youman, C., "Role-based access control models," *IEEE Computer*, vol. 29, no. 2, pp. 38–47, 1996.

[77] Sandhu, R., Ferraiolo, D., and Kuhn, D., "The NIST model for role based access control: Toward a unified standard," *5th ACM Workshop Role-Based Access Control*, pp. 47–63, 2000.

[78] Schneider, F., Walsh, K., and Sirer, E. G., "Nexus authorization logic (nal): Design rationale and applications," tech. rep., Cornell Computing and Information Science, Cornell University, September 2009.

[79] Seamons, K. E., Winslett, M., and Yu, T., "Limiting the disclosure of access control policies during automated trust negotiation," in *Network and Distributed System Security Symposium*, February 2001.

[80] Seigneur, J., Gray, A., and Jensen, C. D., "Trust transfer: Encouraging self-recommendations without sybil attack," in *Third International Conference on Trust Management, LNCS, Springer*, 2005.

[81] Shieh, A., Williams, D., Sirer, E., and Schneider, F., "Nexus: A new operating system for trustworthy computing," in *Symposium on Operating Systems Principles*, 2005.

[82] SecPAL. http://research.microsoft.com/en-us/projects/secpal/ (April 2011).

[83] Squicciarini, A., Bertino, E., Ferrari, E., Paci, F., and Thuraisingham, B., "Pp-trust-x: A system for privacy preserving trust negotiations," *ACM Transactions on Systems and Information Security*, 2007.

[84] Turkmen, F. and Crispo, B., "Performance evaluation of xacml pdp implementations," *Secure Web Services*, 2008.

[85] Twigg, A. and Dimmock, N., "Attack-resistance of computational trust models," *Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 2003.

[86] Wang, L., Wijesekera, D., and Jajodia, S., "A logic-based framework for attribute based access control," in *ACM workshop on Formal Methods in Security Engineering*, 2004.

[87] Winsborough, W. H., Seamons, K. E., and Jones, V. E., "Negotiating disclosure of sensitive credentials," in *2nd Conference on Security in Communication Networks*, 1999.

[88] Xiong, L. and Liu, L., "Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, July 2004.

[89] Yu, H., Kaminsky, M., Gibbons, P. B., and Flaxman, A., "Sybilguard: Defending against Sybil attacks via social networks," *SIGCOMM*, 2006.

[90] Yu, W. and Nayak, E., "An algorithmic approach to authorization rules conflict resolution in software security," in *Annual IEEE International Computer Software and Applications Conference*, July 2008.

[91] Yuan, E. and Tong, J., "Attributed based access control (ABAC) for web services," *Proceedings OF THE IEEE International Conference on Web Services (ICWS)*, 2005.