PRIVACY PROTECTION FOR ROLE-BASED ACCESS CONTROL IN SERVICE ORIENTED ARCHITECTURE

QA
76.9
A25
79
2010

By

Ye Yu

Bachelor of Science

Zhongshan (Sun Yat-sen) University

Guangzhou, China, 510275

1994

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Applied Science

in the Program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2010

©Ye Yu 2010

PROPERTY OF
RYERSON UNIVERSITY LIBRARY

Author's Declaration

I hereby declare that I am the sole author of this thesis and the work was done under the guidance of Professor Cungang Yang, at the Ryerson University.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Author's signature: _____

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Author's signature: _____

Abstract

Privacy Protection For Role-based Access Control in Service Oriented Architecture

© Ye Yu

Master of Applied Science

Electrical and Computer Engineering

Ryerson University

Toronto, 2010

Service Oriented Architecture (SOA) changes the way of conducting business by opening their services/information to the larger business world over the networks. However, properties of “open” and “interoperable” of SOA make privacy security a sensitive issue since this architecture combines unknown individuals and individuals systems together. In SOA, service providers limit the permission of access to specific authorized individuals only, so they have to verify these individuals’ identity information to decide if permission should be granted or not. On the other hand, access requestors are not willing to disclose their privacy in an open system to unknown parties. In this thesis, we present cryptography-based protocols to solve this issue of conflict. Our protocols are proposed for Role-Based Access Control (RBAC), which is a most popular access control methodology, in SOA environment. In addition, our protocols are compatible with current SOA standards and technologies such as XACML and SOAP.

Acknowledgements

I am heartily thankful to my supervisor, Dr. Cungang Yang, whose guidance, encouragement and support from the initial to the final level enabled me to develop an understanding of the subject. During the period of research work, Dr. Yang's truly scientist intuition inspires and enriches my growth as a student and researcher. I am indebted to him more than he knows.

Also, I thank my wife Grace Fen Yin and two lovely children Matthew Yu and Michelle Yu for their sacrifice and support throughout all my studies at University.

Content

Declaration.....	ii
Abstract.....	iii
Acknowledgements	iv
Content.....	v
List of Figures.....	viii
List of Tables	x
1. Introduction	1
1.1 Introduction to Service Oriented Architecture	1
1.2 Introduction to SOA Security	3
1.3 Introduction to Role-Based Access Control	4
1.4 Our Contribution	8
2. Related Works	10
2.1 Non-cryptography Based Trust Negotiation.....	11
2.2 Cryptography Based Privacy Protection Solution for Attributes or Identity Based Access Control.....	12
2.3 Cryptography Based Privacy Protection Solution for RBAC.....	13
3. Preview of Related Cryptographic Technologies	14
3.1 Identity-based Encryption	14
3.2 Homomorphic Encryption	16
3.3 Set Intersection	17
3.4 Oblivious Transfer.....	18
4. Proposed RBAC Solution in SOA.....	19
4.1 SOA Access Control Standard - XACML.....	20
4.1.1 XACML Processing Model	20

4.1.2	XACML Policy Language	22
4.1.3	XACML Request and Response Language.....	23
4.1.4	A Complete Access Control Sample of XACML	23
4.2	RBAC Profile of XACML.....	26
4.2.1	Role PolicySets and Permission PolicySets	26
4.2.2	Attribute-Based Role Assignment	30
4.2.3	Conclusion on RB-XACML	31
4.3	Proposed SOA RBAC Solution.....	32
4.3.1	General Design of Our Proposed SOA RBAC Solution	32
4.3.2	Login & Role Assignment Process:	35
4.3.3	Access Request & Response Process:.....	36
4.4	Privacy Protection Targets for the SOA RBAC Solution.....	39
5.	Privacy Protection Protocols in Login & Role Assignment process.....	40
5.1	Attributes Credentials Values Exchange Stage	43
5.1.1	Attributes Standards and Preparation.....	43
5.1.2	Privacy Protocol – I: Naïve Protocol	44
5.1.3	Privacy Protocol - II: Sophisticated Protocol.....	46
5.1.4	Privacy Protocol - III: Improved Sophisticated Protocol.....	49
5.1.5	Comparison of Three Protocols	51
5.2	Policies Mapping Stage	52
5.2.1	Principles for Policies Mapping and Role Assignment.....	52
5.2.2	Solution One for Policy Mapping	54
5.2.3	Solution Two for Policy Mapping	55
5.2.4	Solution Three for Policies Mapping	57
5.2.5	Comparison of Three Solutions	59
5.2.6	Role Hierarchy and Constraints	60
5.2.7	Role Credential and Role Assignment	61
5.3	Security Review for Role Assignment Process	62

6. Privacy protection in Access Request & Response Process.....	63
6.1 Goals and Principle for Our Privacy Protection Protocols	63
6.2 Our Role Privacy Protection Protocol	65
6.2.1 Privacy Protection Protocol for Information Access Request.....	66
6.2.2 Privacy Protection Protocol for Service Access Request.....	68
6.2.3 Role Hierarchy and Complicated Roles-to-Permission Policies	70
6.3 Implementation of our protocols in Access Request & Response Process.....	73
6.3.1 Implementation of Information Access Request Protocol	73
6.3.2 Implementation of Service Access Request Protocol	78
6.4 Security Review on Protocols for Access Request & Response Process	81
7. Conclusion and Future Work	83
References.....	84
Glossary	88

List of Figures

Figure 1: Basic SOA Model	2
Figure 2: Current SOA Security Standards Framework	3
Figure 3: RBAC model.....	6
Figure 4: Sample of Role Hierarchy.....	7
Figure 5: Example of Credential Exchange Sequence.....	11
Figure 6: IBE System	15
Figure 7: 1-out-of-2 OT.....	19
Figure 8: Traditional SOA XACML based Access Control model	22
Figure 9: XACML Request Sample	24
Figure 10: XACML Policy Sample	25
Figure 11: XACML Response Sample	26
Figure 12: Sample – Permission PolicySet for IT Manager Role.....	27
Figure 13: Sample – Permission PolicySet for Senior Developer Role.....	28
Figure 14: Sample – Role PolicySet for IT Manager Role.....	29
Figure 15: Sample – Role PolicySet for Senior Developer Role.....	30
Figure 16: Attribute-Based Role Assignment.....	31
Figure 17: Simplified SOA Access Control Solution.....	33
Figure 18: Single Sing on	34
Figure 19: Login & Role Assignment Process	35
Figure 20: Access Provider Response Process	37
Figure 21: Sample of Modified Request for SOA RBAC Solution.....	38
Figure 22: Naïve Protocol	45
Figure 23: Sophisticated Protocol.....	47
Figure 24: Improved Sophisticated Protocol	49
Figure 25: Policy Mapping - Solution One.....	54

Figure 26: Policy Mapping - Solution Two	55
Figure 27: Flow Chart for Policy-Role Mapping Process	58
Figure 28: Privacy Protection Protocol for Information Access Request	66
Figure 29: Privacy Protection Protocol for Service Access Request	68
Figure 30: XACML Request In Tradition Way	73
Figure 31: XACML Request under Privacy Mode	74
Figure 32: Sample – Resource PolicySet for developer-guide.doc	75
Figure 33: Sample – Permission PolicySet for developer-guide.doc	76
Figure 34: Sample of Encrypted SOAP Message in Information Access Request	77
Figure 35: Sample – Resource PolicySet for Service	78
Figure 36: Sample of Encrypted SOAP Message to Transfer Encrypted S	79
Figure 37: Sample of Encrypted SOAP Message to Transfer Encrypted {S1, S2, ... Sn}	80
Figure 38: Sample of Response in Service Access Request	80

List of Tables

Table 1: Notation.....	43
Table 2: Comparison of Three Protocol for Value Exchange Stage.....	52
Table 3: Comparison of Three Solutions for Policies Mapping Stage	60

1. Introduction

Service Oriented Architecture (SOA) has become widely popular and acknowledged. The benefits driving the organizations to adopt SOA primarily come down to two factors. First, SOA can provide organization flexibility to be able to change their IT infrastructure faster than before. Second, SOA saves development dollars through reuse of services.

SOA is a distributed system open to a wider world and combines unknown individual systems together. As a result, with the great convenience and success brought by SOA, it also causes security issues. In this thesis, we will propose protocols to protect identity privacy for Role Based Access Control (RBAC) in SOA. This chapter introduces this thesis by describing the background knowledge in SOA, SOA Security Framework and RBAC.

1.1 Introduction to Service Oriented Architecture

There is no authoritative definition for “Service Oriented Architecture (SOA)” yet. Different resources explain SOA in different ways. One of typical definition can be found in [15], “Service Oriented Architecture (SOA) is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains.” However, the basic concept of SOA is the same for all these definitions and we can understand SOA in this way:

- SOA is created to implement a complicated business process;
- The complicated business process is composed of multiple business functions;
- Each business function is implemented by an independent IT system, that means business functions may have different IT environments (Operation System, Database, Programming Language etc.)
- SOA is a structure or solution to combines those different unconnected functions together to realize the integrated business process.

SOA has the following characteristics:

- Concept of SOA is developed from Distributed system which consists of multiple autonomous computer systems that communicate through a computer network (secured or unsecured);

- SOA is loose coupled. Sub-systems under SOA do not communicate with each other directly from database or program level. Instead they communicate through an application interface (services);
- Service is core of SOA. Even though there is no explicit standard, most of SOA is based web Service which is application programming interface (API) expressed in Extensible Markup Language (XML) format.

Now we know that web service is the most important component and the key for SOA. A typical definition of Web Service in [16] is “A Web Service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.” We may also understand Web Service in this way: it is a modular XML-based application using standards and technologies for distributed computing and defined with WSDL, published in UDDI and invoked via SOAP.

Figure 1: Basic SOA Model [15]

Figure1 [15] is a basic SOA model, from which we can know that major elements in SOA are:

- Service Consumer (alternative name: Service Client, Service Requestor, Application Builders, Access Requestor) which applies to access a service;
- Service Provider which provides service;

- Service Broker (alternative name: Service Registry, UDDI – Universal Description Discovery Integration) is a place where service providers publish services information and service consumers query/find needed service.

Among these elements, Access Requestor (AR), and Service Provider (SP) are two of major parties that we will discuss in this thesis.

1.2 Introduction to SOA Security

As we mentioned in Section 1.1, security is becoming major concerns and blockage for SOA popularization and promotion. As a result, SOA security becomes a hot topic for academic research and industry interest [11][18][20][21][24][25][26]. Lots of organizations including W3C, WS-I, OASIS, IT industry giants such as IBM, HP, Oracle, BEA etc. and academic organizations are working on SOA security architecture, standards and protocols. Figure 2 [13] provides us a clear picture about a SOA security standards framework which represents major achievements in SOA security researches and implementations.

Figure 2: Current SOA Security Standards Framework [13]

From Figure 2 [13], we can find the following major security components and their functions in SOA Security Framework:

- Network Level Security (IP Security or IPsec): provides secure sessions with host authentication, data integrity and data confidentiality during the data transportation process in network;
- Transportation Level Security (Secure Socket Layer or SSL): provides authentication, data integrity and data confidentiality;
- XML Security (XML Signature and XML Encryption): provides data integrity and data confidentiality at XML document level;
- SOAP Messages Security (WS Security and WS SecureConversation): provides single SOAP message integrity and confidentiality and mechanisms for associating security tokens with messages content (header and body blocks);
- Access Control & Policies (XACML, SAML and WS-Policy): provides standards for access policies, access request & response and security assertion etc.;
- Identity Management (SAML, Liberty, WS-Federation): provides and promote interoperability between disparate authentication and authorization systems;
- Security Management (XKMS – XML Key Management Standard, WS-trust): provides general security function such as key management and enable the issuance and dissemination of credentials among different trust domains (individual systems).

Figure 2 gives us a clear picture of SOA security framework which has been carefully designed and implemented, and covers most of SOA security concerns. However, this framework does not include Identity Privacy Protection. SOA is loosely coupled by individual systems, and clients are not willing to disclose their identity information (Business Identity or Personal Identity) to unknown systems. The lack of Identity Privacy Protection of current SOA security framework is the main motivation of our research works.

1.3 Introduction to Role-Based Access Control

Access control technology has evolved from research and development efforts supported by the Department of Defense (DoD). This research has resulted in two fundamental types of access control: Discretionary Access Control (DAC) and Mandatory Access Control (MAC). After that,

new access control methodologies were developed to meet requirements of industry organization. Most important access control technologies are listed as following:

- Mandatory Access Control (MAC) or Lattice-based Access Control (LBAC): the operating system constrains the ability of a subject or initiator to access or generally perform some sort of operation on an object or target [17]. MAC mechanisms assign a security level to all information as well as a security clearance to each user, and ensure that all users only have access to that data for which they have a clearance ;
- Discretionary Access Control (DAC): a means of restricting access to objects based on the identity of subjects and/or groups to which they belongs;
- Identity-based Access Control (IBAC) which is an unofficial concept/type in access control, however it is also popularly used. In this model, access permissions and policies are applied to the identity of access Requestors. We can regard IBAC as a kind of DAC;

Under IBAC, access permissions are assigned to individuals based on their identity. For example, a policy of IBAC states: "EDIT permission for code of main.java is assigned to John Smith with employee ID of DEV20100615001".

- Attributes-based Access Control (ABAC): access permissions are decided by attributes of subjects, objects and environment.

For example, under ABAC, a policy states "EDIT permission for code of main.java is assigned to those whose position is developer (subject attribute) during the time period of 8:00AM to 5:0PM (environment attribute)".

- Role-Based Access Control (RBAC), which is the access control approach discussed in this thesis, we will explain it in detail in the chapter;

In [19], with RBAC, access decisions are based on the roles that individual users/subjects have as part of an organization. Users take on assigned roles (such as IT Manager, Senior Developer, Junior Developer). Access rights/privileges are grouped by role name, and the use of resources is restricted to individuals authorized to assume the associated role. RBAC model can be very complicated. In this thesis, we will only study on basic RBAC model. Figure 3 [49] demonstrates a typical RBAC model.

Figure 3: RBAC model [49]

Major components of the RBAC model are as follows:

Users & Roles: For simplicity in this model, a user can be regarded as a human being. A role is a named job function within the organization that describes the authority and responsibility conferred on a member of the role. [49] Actually, range of concept of “User” can be expanded to a function, sub-system or system etc.

Permissions: Permission is an approval of particular mode of access to one or more objects in the system. The terms of authorization, access right, and privilege are also used in the literature to denote permission.

Sessions: Users establish session during which they may activate a subset of roles they belong to. Each session maps one user to possibly many roles.

Role Hierarchies: Role Hierarchies are a natural means for structuring roles to reflect an organization’s lines of authority and responsibility (see Figure 4). By Convention, more powerful (senior) roles are shown toward the top of these diagrams and less powerful (junior) roles are shown toward the bottom. [49] Permissions of junior (child) roles will be automatically inherited by senior (parent) role.

Constraints: Constraints are an important aspect of RBAC and are sometimes argued to be the principle motivation behind RBAC. [49] A common example is that of mutually disjoint organizational roles, such as Developer and QA (a person can not hold a role of developer and

QA at the same time). Constraints can also be placed on (PA) *permission_assignment*. For example, permission of “modifying code” & permission of “publishing code” can not be assigned to a same role.

Figure 4: Sample of Role Hierarchy

Characteristics and concept of RBAC can also be understood as following:

- A user/subject is granted access to an object based on the assigned role;
- Roles are defined based on job functions;
- Permissions are defined based on job authority and responsibilities within a job function;
- Operations on an object are invoked based on the permissions;
- The object is concerned with the user’s Role rather than the user (identity or attributes);

RBAC system enables users to carry out a broad range of authorized operations, and provides great flexibility and breadth of application. System administrators can control access at a level of abstraction (based on job function rather than individual) that is natural to the way that enterprises typically conduct business [19]. RBAC greatly reduces work load for access control task, and makes it flexible and scalable to grow. For distributed systems (such as SOA), RBAC administrator’s responsibilities can be divided among central and local protection domains; that is, central protection policies can be defined at an enterprise level while leaving protection issues that are of local concern at the organizational unit level. In SOA, we can also let role definition

and assignment to be handled at enterprise level, and each unit can freely maintain (create/modify/delete) access policies based on their business requirement and security concerns. (We will discuss this model in details next chapter)

RBAC becomes a promising method for controlling what information computer users can utilize, what program they can run and what modifications that they can make since this methodology was brought up. Actually, RBAC is already popularly used by a large range of IT system and organizations, and it's also a hot topic in academic research.

1.4 Our Contribution

In this thesis, we contribute from the following aspects:

- **Overall Concept and Idea.** Our research works step across fields of SOA, RBAC and privacy protection to provide privacy protection solution for RBAC model in SOA environment. The overall concept, idea and solution elaborated in this thesis, as a whole, is one of the major contributions of this thesis. It is the first academic research in these cross fields, and in addition this research work can be valuable and beneficial to industry since SOA & RBAC are popular architecture and access control methodology accordingly and privacy is a sensitive security issue now.

SOA has a general access control standard - XACML and RBAC standard - RBAC profile of XACML. XACML is a rich language and complicated. So, it is not possible for us to updated XACML to fit our protocols. On the contrary, it's required that our proposed privacy protection protocols should be compatible with current SOA standards so that the protocols can be feasible to be implemented in SOA. Our solution in this thesis meets this requirement and it is compatible with current SOA standards.

Our solution proposed in this thesis is composed of three parts: new SOA RBAC structure (modeling); privacy protection protocols in Login & Role Assignment process; and privacy protection protocols in Access Request & Response process. We will discuss these three parts in details in Chapter 4, Chapter 5, and Chapter 6 accordingly.

- **New SOA RBAC Structure (Modeling).** New SOA RBAC structure introduced in this thesis divides RBAC process into two separate process blocks: Login & Role Assignment Process, and Access Request & Response process. Login & Role Assignment Process is handled by a separate system apart from traditional SOA structure, and Access Request &

Response process however, is still operated within SOA. Login and role assignment process and related operations are handled by three components: Access Request (AR), Role Assigner (RA) and Policy Verifier (PV). These three components are working together to realize role assignment in a privacy preserving manner during login & role assignment process. Separating Login & Role Assignment process from SOA main structure makes it possible for us to propose protocols for a complete privacy protection solution (ZERO Knowledge disclosure) for this process. The separation also has other benefits such as meeting requirements of Single Sign on, and easy system maintenance and policy management etc.

- **Privacy Protection Protocols and Solution in Login & Role Assignment Process.** This process is divided into two stages/operations: attributes value exchange & policy mapping.

We propose three protocols to achieve attributes value exchange operation between Access Requestor and Role Assigner: Naïve Protocol, Sophisticated Protocol and Improved Sophisticated Protocol. All these protocols are cryptography-based and each protocol has its advantage and disadvantage. Naïve Protocol has only two steps. However it causes exchange values explosion which would consume system's memory storage and computation capability and reduce system efficiency. Sophisticated Protocol has 5 steps and it successfully solved values explosion issue. However, Role Assigner will get to know how many valid credentials Access Requestor holds. Improved Sophisticated Protocol is a complete ZERO knowledge disclosure protocol. However, there are six steps involved in this protocol. Organization can choose one of these three protocols based on their situation and priorities.

For policy mapping operation, we compare three solutions for mapping role assignment policies with attributes values. In Solution One, the Access Requestor simply sends her obtained values as value exchange operation to the Role Assigner. This solution is easy in implementation however it is not privacy preserving because Role Assigner can easily know all Access Requestor's valid attributes via her values. Solution two is cryptography-based and handled at Access Requestor side. This solution discloses some of Role Assigner's policies privacy such as which values are required for a role and how many credentials the role requires etc. In addition, this solution will cause lots of computation at Access Requestor side and requires Access Requestor (client side device) to have strong computation capability. So, it's not an ideal solution. Solution three takes use of a newly introduced 3rd party - Policy Verifier and it is a complete privacy preserving solution for Role Assigner's attributes-to-role policies with ZERO policies knowledge disclosure. This solution is recommended.

- **Privacy Protection Protocols and Solutions in Access Request & Response Process.** In this thesis, we make a thorough analysis on privacy importance and sensitivity for Access Request & Response process, and conclude that privacy protection is not a critical task in this process comparing with Login & Role Assignment process. However, it's still desirable to setup privacy protection mode in case access request need to protect her role information as privacy. So, in our protocol, there is an option for Access Requestor to choose to use "privacy mode" or normal access mode when she submits her access requests.

Proposing Privacy Protection Protocols in Access Request & Response process is more difficult than to proposing Privacy Protection Protocols in Login & Role Assignment process because we need to take into consideration of system efficiency (too complicated cryptography algorithm will slow down the whole process and system); system integration and compatibility (the protocol should be able to implemented using current SOA standards and technologies.). In addition, there are two kinds of access request: information access (required information can be sent to access requestor) and service access (access requestor needs permission of access to a specific function, service or webpage). Protocols for these two kinds of access request are different. We introduce two sets of protocols for these two access requests. They are not ZERO knowledge disclosure protocols. However, it greatly improves privacy protection comparing with traditional SOA access control technology without involvement of complicated cryptography computation. Moreover, the solution is able to be implemented based on current SOA standards such as XACML, SOAP etc. so that our protocols are practical and feasible to be implemented.

2. Related Works

Our research and original ideas come from area of automated trust negotiation. Purpose of trust negotiation is to enable stranger access sensitive data in open environment without violating access policies and privacy. Our research works evolve from original idea of trust negotiation, and step across fields of privacy protection, RBAC and SOA. There is no similar academic research on this area yet, however, there are lots of previous research works which focus on one of related fields in trust negotiation, privacy protection or privacy protection for RBAC, which are suitable for a specific scenario. Based on focus of these research works, we can group them into the following three catalogs:

- Non-cryptography based trust negotiation;

- Cryptography Based Privacy Protection Solution for Attributes or Identity Based Access Control;
- Cryptography Based Privacy Protection Solution for RBAC

We will discuss these three catalogs in the following three sections:

2.1 Non-cryptography Based Trust Negotiation

T. Yu et al. proposed “interoperable strategies for automated trust negotiation” [1]. Their solutions are provided to protect sensitive credentials and services with (access control) policies and establish trust incrementally through a sequence of credential disclosures: begin with credentials that are less sensitive; then build up trust so that more sensitive credentials can be disclosed;

Figure 5: Example of Credential Exchange Sequence

Figure 5 [1] illustrates a trust negotiation sample in T. Yu’s paper. We do not need to understand the terms of CPN and BBB member. We just need to know the process: CPN provides the Landscape Designer less sensitive information “BBB-Member”. First, if she gets a response from Landscape Designer and receive “Credit_Card Reseller_License”, then she will send Landscape

Designer more sensitive information “Order_OK”. “Order_OK $\leftarrow$ (Credit_Card $\vee$ CPN_Account) $\wedge$ Reseller_License” represents kind of complicated policy condition operation.

In this paper, they created sophisticated information exchange model to built trust negotiation. However, defects of their solution are obvious:

- It's not cryptography level solution, and it's easy for attacker to find security leak from their proposed model (for example, is so called “less sensitive” information really not sensitive?).
- It is not able to handle situation of dead cycling where both sides of information exchanging are not willing to disclose information first.
- This solution is not manageable because different individuals have different standard to measure “sensitive” information, as a result there will be heavy management and maintenance work for their counter part to handle many different standards.

2.2 Cryptography Based Privacy Protection Solution for Attributes or Identity Based Access Control

W. Du et al. created “Oblivious Signature-Based Envelope” [2] which is a cryptography-based solution and perfectly solved dead-cycling issue which [1] can not fix.

For example, the sender has a message to report, but she wants to make sure only CIA can receive it. So she wants the receiver to show her the CIA certificate. On the other hand, the receiver won't show his CIA certificate to the sender because that's dangerous, and he just requires the sender send him the message. It is a conflicting situation, and communication won't continue.

Oblivious Signature-Based Envelope is introduced to solve this kind of problem. The basic idea is as follows: sender encrypts the message in an “oblivious signature-based envelope”, so that receiver can open the envelope if and only if he has the required certificate; on the other hand, the sender do not know whether the receiver has the certificate. Then, the communication is accomplished by a privacy friendly manner which can satisfy both receiver and sender.

Oblivious Signature-Based Envelope (OSBE) can be built on Identity-Base Encryption (IBE) technology. Where information sender encrypts sensitive information using received ID as public key, and only receiver can decrypt it using private key from a 3rd party CA (PKG), and it can also use RSA signature to negotiate public key and private key between sender and receiver.

This solution is created to protect privacy for identity-based access control, and it can only be used in some simple situation (single sender, single receiver and ID based IBE). This solution absolutely can not meet our requirement.

K. Frikken et al. introduced “Attribute-Based Access Control with Hidden Policies and Hidden Credentials” [4] which discussed attribute and policy privacy protection in a more complicated access control environment. There are two primary phases in their protocols: a credential hiding phase; and a blinded policy evaluation phase. So, their protocols also provide complete privacy-preserving solution for both access requestor’s privacy and information provider’s policies. They proposed several protocols with different complexity trading-off different security level for each phase. Their protocols are able to efficiently protect attributes privacy of access requestor. However, they can not efficiently protect information provider’s access policies even though they proposed an even more complicated cryptography algorithm. There is no absolute policy privacy security in the case of direct value/trust negotiation between access requestor and service/data provider. If policy mapping is handled at service/data provider side, provider will know which values (which represent credentials/attributes held by access requestor) the access requestor holds; likewise, if policy mapping is handled at access requestor’s side, the requestor will get to know what values are related to specific role assignment policy. There is no policy privacy security in either way without help of 3rd party agent who will perform policy mapping. The 3rd party is trusted by both access requestor and data provider. That’s the reason why we introduce “Policy Verifier” in our proposed model in Chapter 4. In addition, their solution is for attribute-based access control and they are not used in a distributed system like SOA.

J. Li et al. initiated “OACerts: Oblivious Attribute Certificates” [3] which is actually combination of [2] and [4]. It took use of similar technologies used in [4] to expand idea of [2] being able to be utilized in more complicated situation (multiple attributes and policies requirements are involved), and it had the same shortcoming of [4]: policies protection issue and complexity. Also, it’s used for attributes-based access control rather than RBAC.

2.3 Cryptography Based Privacy Protection Solution for RBAC

D. Yao et al. proposed “Compact and Anonymous Role-Based Authorization Chain” [5] which discussed privacy protection in RBAC scenario. This paper provided privacy protection solution for “Role Privilege Delegation” scenarios. For example, a staff of a company with “IT Manager” role can pass the role privileges to a contractor without revealing his identity information. The

solution can only be suitable in specific “Role Privileges Delegation” scenarios. Also, the solution is inefficient because of its one time key and signing permit (every delegation process needs a new one time key and signing permit).

In conclusion, all these five protocols/solutions are all innovative and have success in some particular situations. Actually, they can only be applied within those particular situations. K. Frikken et al. and J.Li’s protocols are perfect for attributes-based access control; W. Du et al.’s work is suitable for identity-based access control; T. Yu et al.’s solution is created for non-cryptography trust negotiation and D. Yao’s work is used only for role privileges delegation. None of these solutions is suitable for our situation – SOA RBAC model. We want to create efficient and flexible identity privacy protection protocol which matches RBAC model and is compatible to current SOA standards. That’s the goal of our research.

3. Preview of Related Cryptographic Technologies

Our privacy protection protocols are cryptography-based, and they are taking use of some existing applied cryptographic technologies. In this chapter, we will introduce several major applied cryptographic technologies which we use in our protocols:

3.1 Identity-based Encryption

Identity-based Encryption (IBE) [2][4][6][37][38] is the most important applied cryptography technology used in this thesis. An IBE public key encryption scheme is a public key system with the added twist that any string can function as a public key. In such system there is a third party that has a secret master-key that enable it to generate the private key corresponding to any public key string. So, the purpose of introducing IBE is to make a string (such as a personal ID, email address, a title of a Role or any other identity-related string) as a public key to encrypt information. Then the sender can encrypt information by using receiver’s ID as public key so that the sender voids obtaining a public key from sender or a directory or another party. Only the right receiver holds private key which is related to her ID and issued by 3rd party private key generator. The private key is kind of credential which can be regarded as approval of receiver’s ID. As a result, no one else except the right receiver can decrypt the message.

The concept of IBE was first proposed by Shamir [37] in 1984; however the first usable IBE systems were proposed recently by Boneh and Cocks [2] [4] [6] [38]. An IBE scheme is specified by the following four steps [4]:

1. Setup: A Private Key Generator (PKG) takes a security parameter k and generates system parameters p and a master secret s . p is public, whereas s is private to PKG;
2. Extract: Given any arbitrary string ID (such as sample@users.example.com , “IT Developer” etc.), PKG uses p , s , and ID to compute the corresponding private key dID;
3. Encrypt: It takes p , ID, and plaintext M as input and returns ciphertext C . We can elaborate the process by a cryptography equation: $C = I(M, ID)$;
4. Decrypt: It takes p , dID, and ciphertext C as input and returns the corresponding plaintext M . The according cryptography equation is $M = I^{-1}(C, PID)$;

After setup, message sender holds system parameter p , arbitrary string ID (plain text definition of target receiver) which are public information. Then she can encrypt her plaintext M by p and ID. Receiver holds public parameter p and dID as private key, and she can decrypt the encrypted message by p and dID. The other person without dID can not decrypted the message even if they obtain ciphertext by whatever method.

Figure 6: IBE System

Figure 6 is a typical IBE system. In short, in IBE, Identities (or Attributes or any recognized string) = Public Keys, and Sender uses the Public Keys to encrypt sensitive message. Receiver

receives corresponding private key from PKG, and decrypt the encrypted message. One of advantage of IBE is its ease to be used, and there is no certificate management; no revocation lists; and no pre-enrollment;

IBE is an important technology we used in this thesis. First, it's used at login and role assignment process by RA who uses her attribute as public key and RA who uses her credentials (obtained from 3rd parity CA who performs as PKG in this case) as private key. IBE is also used at RBAC based information access stage where information providers use Role's title as public key to encrypt information and AR uses Roles' credentials (obtained from RA who performs as PKG in this case at login process) as private key to decrypt information. Details will be discussed in Chapter 5 and Chapter 6.

3.2 Homomorphic Encryption

Homomorphic encryption is a form of encryption where specific algebraic operation is performed on the plaintext and a possibly different algebraic operation is performed on the ciphertext. Depending on one's viewpoint, this can be seen as either a positive or negative attribute of the cryptosystem. Homomorphic encryption schemes are malleable by design. The homomorphic property of various cryptosystems can be used to create secure voting systems, collision-resistant hash functions, and private information retrieval schemes. [17]

For example, a cryptographic scheme is said to be homomorphic if, for its encryption function E , the following holds: $E(x) \cdot E(y) = E(x + y)$. This encryption function is exactly what we will use in our protocol.

Homomorphic can be realized by different cryptosystems, and here are four samples:

By Unpadded RSA [17]:

If the RSA public key is modulus m and exponent e , then the encryption of message x is give by $E(x) = x^e \bmod m$.

The homomorphic property is then:

$$E(x_1) \cdot E(x_2) = x_1^e \cdot x_2^e \bmod m = (x_1 \cdot x_2)^e \bmod m = E(x_1 \cdot x_2)$$

By ElGamal [17]

In the ElGamal cryptosystem, in a group G , if the public key is (G, q, g, h) , where $h = g^x$, and x is the secret key, then the encryption of a message m is $E(m) = (g^r, m \cdot h^r)$, for some $r \in \{0, 1, \dots, q-1\}$. The homomorphic property is then:

$$E(x_1) \cdot E(x_2) = (g^{r_1}, x_1 \cdot h^{r_1}) \cdot (g^{r_2}, x_2 \cdot h^{r_2}) = (g^{r_1+r_2}, (x_1 \cdot x_2) \cdot h^{r_1+r_2}) = E(x_1 \cdot x_2)$$

By Benaloh [17]

In the Benaloh cryptosystem, if the public key is the modulus m and the base g with a blocksize of r , then the encryption of a message x is $g^x u^r \pmod m$. The homomorphic property is then:

$$E(x_1) \cdot E(x_2) = (g^{x_1} u_1^r) \cdot (g^{x_2} u_2^r) = g^{x_1+x_2} (u_1 \cdot u_2)^r = E(x_1 + x_2) \pmod m$$

By Paillier [17]

In the Paillier cryptosystem, if the public key is the modulus m and the base g , then the encryption of a message x is $E(x) = g^x r^m \pmod{m^2}$. the homomorphic property is then:

$$E(x_1) \cdot E(x_2) = (g^{x_1} r_1^m) \cdot (g^{x_2} r_2^m) = g^{x_1+x_2} (r_1 \cdot r_2)^m = E(x_1 + x_2) \pmod{m^2}$$

These are just some samples cryptosystems for homomorphic. Actually, there are more homomorphic cryptosystems. More homomorphic schemes are discussed in [40][41][42][4]. A homomorphic encryption scheme is semantically secure if $E[x]$ reveals no information about x ; hence, from a pair $(E(x_1), E(x_2))$, it is computationally infeasible to distinguish between the case $x_1 \neq x_2$ and $x_1 = x_2$.

3.3 Set Intersection

[43][4] introduces a perfect scheme for calculating intersection of two k element sets. In this paper, the following protocol is proposed:

[43]There are two sides for information exchange, let's identify them as C (Client) and S (Server):

Client (C) defines a polynomial of degree k whose roots are his inputs $x_1, \dots, x_k$

$$P(y) = (x_1 - y)(x_2 - y) \dots (x_k - y) = a_0 + a_1 y + \dots + a_k y^k$$

C sends to server S homomorphic encryptions of polynomial's coefficients

$$Enc(a_0), \dots, Enc(a_k)$$

$$Enc(P(y)) = Enc(a_0 + a_1 \cdot y^1 + \dots + a_k \cdot y^k) = Enc(a_0) \cdot Enc(a_1) y^1 \cdot \dots \cdot Enc(a_k) y^k$$

S uses homomorphic properties (Section 3.2) to compute:

$$Enc(\text{random} \cdot P(y) + y)$$

If $y \in X$, $X = \{x_1, x_2, \dots, x_k\}$, we can see that $P(y) = 0$, so,

$\text{Enc}(\text{random} \cdot P(y) + y) = \text{Enc}(y);$

Otherwise, $P(y) \neq 0$, so,

$\text{Enc}(\text{random} \cdot P(y) + y) = \text{Enc}(\text{random});$

From the variant protocols, the formular can be updated to:

$\text{Enc}(\text{random} \cdot P(y) + 0)$

So, If $y \in X$, $X = \{x_1, x_2, \dots, x_k\}$,

$\text{Enc}(\text{random} \cdot P(y) + 0) = \text{Enc}(0)$

Otherwise, $\text{Enc}(\text{random} \cdot P(y) + 0) = \text{Enc}(\text{random})$

From the paper, we can conclude this valuable function: $\text{SETINT}(x, S, EA)$, where one side (for example RA) provides a value x , and the other side (for example AR) provides a set of values. EA is a semantically secure homomorphic encryptions system.

The result of the function is that RA will learn $EA[y]$ where y is 0 if and only if $x \in S$, and otherwise y is a random number.

3.4 Oblivious Transfer

In cryptography, an Oblivious Transfer protocol (OT) [33] [34] [35] [36] is a protocol by which a sender sends some information to the receiver, but remains oblivious as to what is received. We can think of a following situation: we have a sender and receiver, where sender has a list of n strings $x_1, \dots, x_n$, and the receiver wants to learn x_i . Of course, the client can simply send i to the sender, but the receiver does not want the sender know the number of “ i ”. Now, the sender just simply send the n strings to the receiver but the sender does not want the receiver to learn all other strings rather than x_i . That is receiver will not learn x_j where $j \neq i$.

OT mainly solves the problem in this situation. With OT technology, on the one hand the receiver will only know x_i , and not all other strings x_j ($j \neq i$); on the other hand, the sender will not know the “ i ”, that is sender will not know which string the receiver takes.

There are three popular OT technologies:

- Rabin’s oblivious transfer protocol;
- 1-out-of-2 obvious transfer
- 1-out-of- n or k -out-of- n oblivious transfer

This thesis will make use of 1-out-of-2 oblivious transfer technology, and we will describe it in details:

In 1-out-of-2 OT situation, the sender sends an ordered pair of bits (b_0, b_1) to a 1-2-OT machine. The receiver then gives the machines a bit i ($i \in \{0, 1\}$), indicating which input he would like to receive. The machine outputs b_i and discards b_{1-i} . Sender knows that Receiver has one of the bits, but not which one.

Figure 7: 1-out-of-2 OT [33]

4. Proposed RBAC Solution in SOA

SOA is distributed system which is composed of individual sub-systems of different technology backgrounds and access control policies. Nature of distributed system makes access control in SOA more challenging than that in centralized systems because:

- There is no boundaries of composed/unit systems;
- There is no prior relationship between access requestor and service provider;
- Subjects (identities or roles of access requestors) and attributes should be unified over the range of SOA;
- Each individual system should be autonomous in maintaining her access control policies which applied to unified subjects;

To handle the above problems or challenges, OASIS standards organization is keeping working on to build and improve SOA access control standard – XACML which is popularly used by industry. Our proposed access control solution is also compatible with this standard to make implementation feasible and practical.

Identity-Based Access Control and Attribute-Based Access Control were normally used as access control approach in practice of SOA implementation. In this situation, access permission is granted based on access requestors' identity (asserted by SAML) or attributes. However, due to

its weak points of being un-scalable (colossal policies pool), not manageable and not-privacy-friendly, people are working on a solution with better access control method. Academic researchers and industry begin to take use of a more flexible and scalable access control methodology – Role-Based Access Control as access control technology [8][9][10][11][12][27] for SOA. RBAC solves “scalable” issue, and emerging of “RBAC profiles of XACML”[12] makes RBAC compatible with XACML. However, RBAC profile of XACML is still not a privacy preserving standards, and this is the reason why we should to create a new RBAC protocols for SOA.

4.1 SOA Access Control Standard - XACML

To propose our RBAC model for SOA, we need to understand SOA access control standard - XACML first. In SOA, XACML [12][13][14][30][31][32][44][45][46] stands for *eXtensible Access Control Markup Language*. It is a declarative access control policy language and access control decision request/response language implemented in XML and a processing model, describing how to interpret the policies and decide access permissions. In short, XACML includes the three main components: Policy Language (in XML); Access control decision request/response language (in XML); and Processing model.

4.1.1 XACML Processing Model

Figure 8 [12] illustrates a traditional SOA access control model [12][18][30].Major components/terms in the model are listed as following:

- PEP (Policy Enforcement Point): Resource specific authorization decision request/response handling and policy defined obligations execution;
- PDP (Policy Decision Point): Evaluates the authorization request against the policy defined for a particular job, resource and user attributes/roles;
- PAP (Policy Authority Point) or Policy DB: policy storage (distributed);
- PIP (Policy Information Point): Supply external policy context and attributes (subject credentials and attributes verification);
- RIP (Resource Information Point): Provide resource context.
- AA (Attribute Authority): Manage User attributes.

From Figure 8 below, we know the XACML Data Flow as follows:

1. PAP: policies/sets → PDP
2. Access Request sends request to PEP
3. PEP sends request to context handler in its native request format, optionally including attributes of the subjects, resources, action and environment
4. context handler constructs an XACML request context and sends it to the PDP
5. PDP requests any additional subject, resources, action and environment attributes from the context handler
6. Context handler requests attributes from PIP
7. PIP obtains the requested attributes.
8. PIP returns requested attributes to the context handler
9. Optionally, the context handler includes the resource in the context
10. Context handler sends requested attributes and (optionally) the resource to the PDP. PDP evaluates the policy
11. PDP returns response context (including the authorization decision) to the context handler
12. Context handler translates response context to the native response format of the PEP. Context handler returns the response to the PEP
13. PEP fulfills the obligations.

The data flow model clearly illustrates that traditional SOA access control is an attributes-based access control. It has serious police volume and privacy issue – that is exactly the motivation of our research and motivation.

Figure 8: Traditional SOA XACML based Access Control model [12]

4.1.2 XACML Policy Language

At the root of all XACML policies is a Policy or a PolicySet. A PolicySet is a container that can hold other Policies or PolicySets, as well as references to policies found in remote locations. A Policy represents a single access control policy, expressed through a set of Rules. Each XACML policy document contains exactly one Policy or PolicySet root XML tag.

Because a Policy or PolicySet may contain multiple policies or Rules, each of which may evaluate to different access control decisions, XACML needs some way of reconciling the decisions each makes. This is done through a collection of Combining Algorithms [12]. Each algorithm represents a different way of combining multiple decisions into a single decision. There

are Policy Combining Algorithms (used by PolicySet) and Rule Combining Algorithms (used by Policy). An example of these is the Deny Overrides Algorithm, which says that no matter what, if any evaluation returns Deny, or no evaluation permits, then the final result is also Deny. These Combining Algorithms are used to build up increasingly complex policies, and they are what allow XACML policies to be distributed and decentralized. While there are several standard algorithms, you can build your own to suit your needs. [48]

4.1.3 XACML Request and Response Language

In addition to define a standard format for policy, XACML defines a standard way of expressing Requests and Responses. A Request contains Attributes in each of the four categories: Subject, Resource, Action, and Environment (which is optional). There is exactly one collection of Resource and Action Attributes in a Request, and at most one collection of Environment Attributes. There may be multiple collections of Subject Attributes, and each collection is identified by a category URI (or, if no category is specified, then the default category is used). In addition to Attributes, the Resource section allows the inclusion of the content of the requested resource, which can be considered in policy evaluation through XPath expressions. The only Attribute that is required in a Request is the Resource identifier.

A Response consists of one or more Results, each of which represents the result of an evaluation. Multiple Results can only be caused by evaluation of a hierarchical resource (explained later), so typically there will only be one Result in a Response. Each Result contains a Decision (Permit, Deny, NotApplicable, or Indeterminate), some Status information (for instance, why evaluation failed), and optionally one or more Obligations (things that the PEP is obligated to do before granting or denying access). The Request and the Response provide a standard format for interacting with a PDP.

4.1.4 A Complete Access Control Sample of XACML

The scenario shows a user trying to access a web page. The request includes the Subject's identity as well as a group they belong to, the resource being accessed, and the action being taken. The policy applies to anyone taking any action on the resource, and contains a rule that applies to a specific action. The Condition in the Rule requires certain group membership to access the resource.

The Request:

```
<Request>
  <Subject>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
      DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name">
      <AttributeValue>sample@users.example.com</AttributeValue> </Attribute>
    <Attribute AttributeId="group" DataType="http://www.w3.org/2001/XMLSchema#string"
      Issuer="admin@users.example.com">
      <AttributeValue>developers</AttributeValue></Attribute>
    </Subject>
  <Resource>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
      DataType="http://www.w3.org/2001/XMLSchema#anyURI">
      <AttributeValue>http://server.example.com/code/docs/developer-
guide.html</AttributeValue></Attribute>
    </Resource>
  <Action>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
      DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>read</AttributeValue></Attribute>
    </Action>
  </Request>
```

Figure 9: XACML Request Sample

Figure 9 provides us a typical XACML sample request statement. There are three key types of information included in the code:

- Attributes of subject: in this sample there are two attributes: sample@user.example.com and “developers”. Please be noted, the “developers” is an attribute of the subject (in this case, developers is the position attribute of subject) rather than a role;
- Requested resource: in this sample, it’s a html page:
<http://server.example.com/code/docs/developerperguide.html>
- Requested action/operation: in this sample, it’s “read”.

XACML is a rich language which can be very complicated. Its policy language can hold PolicySet -> Policy -> Rule with different override algorithm. We will only describe it in a simple sample. Figure 10 below is a typical simplified policy language code, and there are also three key components in this sample code: 1. for the subject with a attribute of “developers”; 2. for the requested target resource of: <http://server.example.com/code/docs/developerperguide.html> ; 3. requested action is : “read” .

The Policy:

```
<Policy PolicyId="ExamplePolicy"
  RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-
combining-algorithm:permit-overrides">
  <Target>
    <Subjects><AnySubject/></Subjects>
    <Resources>
      <Resource>
        <ResourceMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#anyURI">http://server.exam
ple.com/code/docs/developer-guide.html</AttributeValue>
          <ResourceAttributeDesignator
DataType="http://www.w3.org/2001/XMLSchema#anyURI"
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
        </ResourceMatch></Resource>
      </Resources>
    <Actions><AnyAction/></Actions>
  </Target>
  <Rule RuleId="ReadRule" Effect="Permit">
    <Target>
      <Subjects><AnySubject/></Subjects>
      <Resources><AnyResource/></Resources>
      <Actions>
        <Action>
          <ActionMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">read</AttributeVal
ue>
            <ActionAttributeDesignator
DataType="http://www.w3.org/2001/XMLSchema#string"
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"/>
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
    <Condition
FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <Apply
FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-
only">
        <SubjectAttributeDesignator
DataType="http://www.w3.org/2001/XMLSchema#string"
AttributeId="group"/>
      </Apply>
      <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">developers</Attrib
uteValue>
    </Condition>
  </Rule>
</Policy>
```

The Response:

```
<Response>
  <Result>
    <Decision>Permit</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
    </Status>
  </Result>
</Response>
```

Figure 11: XACML Response Sample

Figure 11 is the sample of XACML response statement. The decision on the request can either be “Permit” or “Deny”. There are only two keywords of decision in the code: “Permit” or “Deny”.

Service Provider will check access request (Figure 9) based on target policy (Figure 10), and since this request meets requirement defined in the policy, so the decision in response statement (Figure 11) is Permit.

4.2 RBAC Profile of XACML

RB-XACML 2.0 [12][47] was approved as OASIS committee draft on 30 September, 2004. It's standard to implement the core and hierarchical components of ANSI standard in SOA including Roles and Role Hierarchies; Permission-role Assignment Relation and User-role Assignment Relation.

4.2.1 Role PolicySets and Permission PolicySets

According to RBAC profile of XACML, PolicySets is divided into two types: Role PolicySets and Permission PolicySets. In these PolicySets, Role is regarded as a specific XACML Attribute.

Role PolicySets:

Role PolicySets are applicable only to holders of a given Role and they reference the Permission Policy Set that specifies the permissions to be granted to holders of this Role;

Permission PolicySets

Permission PolicySets are never used as base policies, but only by reference from Role PolicySets and are application to any Subject. Permission PolicySets specify a set of Permissions to be granted to holders of a given Role or to holders of any Role that is senior to the given Role.

Here is an example. There are two roles "IT Manager" and "Senior Developer", and they are expressed as two separate values for a single XACML Attribute called "Role". A "Senior Developer" role has permission to "Modify Code", and an "IT Manager" role has permission to

```
<PolicySet PolicySetId="Permission PolicySet for IT Manager Role"
  CombiningAlgorithm="permit-overrides">
  <Target>
    <Subjects>
      <AnySubject/>
    </Subject>
    <Resources>
      <AnyResource/>
    </Resources>
    <Actions><AnyAction/></Actions>
  </Target>
  <Policy PolicyId="Permissions specifically for the IT Manager
role" CombiningAlgorithm="permit-overrides">
    <Target>
      <Subjects><AnySubject/></Subject>
      <Resources><AnyResource/></Resources>
      <Actions><AnyAction/></Actions>
    </Target>
    <Rule RuleId="Permission to Publish Modified Code"
      Effect="Permit">
      <Target>
        <Subjects>
          <AnySubject/>
        </Subjects>
        <Resources>
          <Resource>
            <ResourceMatch MatchId="string-match">
              <AttributeValue>Delete Code</AttributeValue>
              <ResourceAttributeDesignator AttributeId="resource-id"/>
            </ResourceMatch>
          </Resource>
        </Resources>
        <Actions>
          <Action>
            <ActionMatch MatchId="string-match">
              <AttributeValue>Publish</AttributeValue>
              <ActionAttributeDesignator AttributeId="action-id"/>
            </ActionMatch>
          </Action>
        </Actions>
      </Target>
    </Rule>
  </Policy>
  <PolicySetIdReference>Permission PolicySet for Senior Developer
Role</PolicySetIdReference>
</PolicySet>
```

Figure 12: Sample – Permission PolicySet for IT Manager Role

“Publish Modified Code” plus any permission granted to “Senior Developer” role. (“IT Manager” is a more senior role than “Senior Developer”) Now, let’s see how to implement the sample by Permission PolicySets and Role PolicySets:

```
<PolicySet PolicySetId="Permission PolicySet for Senior Developer
Role"
    CombiningAlgorithm="permit-overrides">
    <Target>
        <Subjects>
            <AnySubject/>
        </Subject>
        <Resources>
            <AnyResource/>
        </Resources>
        <Actions><AnyAction/></Actions>
    </Target>
    <Policy PolicyId="Permissions specifically for the Senior
Developer role", CombiningAlgorithm="permit-overrides">
        <Target>
            <Subjects><AnySubject/></Subject>
            <Resources><AnyResource/></Resources>
            <Actions><AnyAction/></Actions>
        </Target>
        <Rule RuleId="Permission to Modify Code" Effect="Permit">
            <Target>
                <Subjects>
                    <AnySubject/>
                </Subjects>
                <Resources>
                    <Resource>
                        <ResourceMatch MatchId="string-match">
                            <AttributeValue>Code</AttributeValue>
                            <ResourceAttributeDesignator AttributeId="resource-id"/>
                        </ResourceMatch>
                    </Resource>
                </Resources>
                <Actions>
                    <Action>
                        <ActionMatch MatchId="string-match">
                            <AttributeValue>Modify</AttributeValue>
                            <ActionAttributeDesignator AttributeId="action-id"/>
                        </ActionMatch>
                    </Action>
                </Actions>
            </Target>
        </Rule>
    </Policy>
    <!-- PolicySetIdReferences can be added here if the Senior
Developer role later obtains permissions associated with any other
role such as Junior Developer -->
</PolicySet>
```

Figure 13: Sample – Permission PolicySet for Senior Developer Role

Figure 12 is sample Permission PolicySet for IT Manager Role. From the bold key words in this PolicySet, we can see that it is to grant “Publish Modified Code” to an IT Manager role and IT Manager role can inherit all permission from Senior Developer role.

Figure 13 is sample RBAC XACML language code for “Permission PolicySet for Senior Developer Role”. The Permission PolicySet grant “Modify Code” permission to Senior Developer role, and it can be still flexible to let Senior Developer to inherit any more junior role such as Junior Developer.

```
<PolicySet PolicySetId="Role PolicySet for IT Manager Role"
  CombiningAlgorithm="permit-overrides">
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="string-match">
          <AttributeValue>IT Manager</AttributeValue>
          <SubjectAttributeDesignator AttributeId="Role"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
    <Resources>
      <AnyResource/>
    </Resources>
    <Actions>
      <AnyAction/>
    </Actions>
  </Target>
  <PolicySetIdReference>Permission PolicySet for IT Manager
  Role</PolicySetIdReference>
</PolicySet>
```

Figure 14: Sample – Role PolicySet for IT Manager Role

Figure 14 is a sample RBAC profile XACML Role PolicySet statement for IT Manager Role. The code provide the following rule: for an access requestor (Subject) with a role of “IT Manager”, check “Permission PolicySet for IT Manager Role” to decide to grant permission or not.

Role PolicySet for Senior Developer is similar to IT Manager Role. Figure 15 is Senior Developer’s Role PolicySet. Based on the Role PolicySet, for all access requestor holding a “Senior Developer” role, their permission decision will be referred to “Permission PolicySet for Senior Developer Role”.

```

<PolicySet PolicySetId="Role PolicySet for Senior Developer Role"
  CombiningAlgorithm="permit-overrides">
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="string-match">
          <AttributeValue>Senior Developer</AttributeValue>
          <SubjectAttributeDesignator AttributeId="Role"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
    <Resources>
      <AnyResource/>
    </Resources>
    <Actions>
      <AnyAction/>
    </Actions>
    </Target>
    <PolicySetIdReference>Permission PolicySet for Senior Developer
    Role</PolicySetIdReference>
  </PolicySet>

```

Figure 15: Sample – Role PolicySet for Senior Developer Role

4.2.2 Attribute-Based Role Assignment

From Figure 16 [12] below, we know that Role Assignment in RB-XACML is based on Subjects (Access Requestor)’s attributes, so we call it “Attributes-Based Role Assignment”. The role assignment is handled at access request process. PDP will evaluate access requestor’s attributes and grant suitable role(s) to her by Role Assignment PolicySets. Then the access request will be evaluated by Role PolicySets and Permission PolicySets based on requestor’s role. Attribute-Based Role Assignment is composed of the following four steps:

Step 1: PDP evaluates Access Requestor’s attributes to decide which role (s) should be assigned to her;

Step 2: After role assignment, assigned role will show up at access request XACML statement/code just like a regular attribute, then, service provider will check Role PolicySet based on the assigned role;

Step 3: Role PolicySet will refer suitable Permission Assignment PolicySet;

Step 4: Permission Assignment PolicySet will check permission rules based on target information (role, target resource and required action etc.) to decide to permit or deny.

Figure 16. Attribute-Based Role Assignment [12]

4.2.3 Conclusion on RB-XACML

From Sections of 4.2.1 and 4.2.2, we can know that RB-XACML is actually a modified version of traditional XACML, which let XACML policy meet features of RBAC (such as Roles definition, permission to role assignments etc.) Our proposed privacy-friendly RBAC proposal for SOA still takes use of current XACML and RB-XACML standards plus some necessary modifications, and these modifications will not cause major change for SOA structure and XACML standards. So, there will be no compatible and implementation issue.

Even though, RB-XACML is created as a standard for RBAC in SOA, we still need to propose a new RBAC solution for SOA because RB-XACML has the following defects:

- Based on RB-XACML, the role is assigned during the access request process. So, each access request will cause a role assignment process. That reduces system efficiency;
- There is no authoritative and recognized organization which provides unified Roles definitions, attributes definitions and attributes-to-roles assignment rules, as a result it will cause lots of coordinate work and maintenance work for a distributed system;
- The role-user assignment is still attribute-based. Furthermore, subject's attributes are still transported between different sub-systems/components to support the process. So, it is not-privacy-friendly. (this is the main motivation of this thesis)
- Role exists as an "Attribute Value" during the process, and it is not a privacy friendly method (in some case, access requestor does not want her role disclosed) because the role of access requestor will be known by counterparts during the process.

Our new SOA RBAC protocol/solution is proposed to solve these defects. The major advantages of our new SOA RBAC solution are listed as the following:

- To realize one role assignment process, then access requestor can use assigned role for all access request within the same session;
- To unify role assignment policies, the policies are agreed and used by all individual systems within SOA range and individual systems are not necessary maintain their own policies;
- To protect attributes privacy, there is ZERO attributes disclosed during the process;
- To protect role privacy if necessary.

4.3 Proposed SOA RBAC Solution

In this section, we will propose our SOA RBAC solution which will conquer the following three major challenges: manageable role assignment policies; privacy protection; and compatible with current SOA standard and technologies such as XACML.

4.3.1 General Design of Our Proposed SOA RBAC Solution

One of major innovations in our proposed SOA RBAC solution is to separate role assignment from traditional SOA structure, and it will be accomplished at login stage before sending access request to an individual system for a specific resource within SOA.

In our proposed solution, the access process is divided into two separate processes: Login & Role Assignment Process – Before Sending Access Request; and Access Request & Response Process – After Receiving Access Request.

Figure 17: Simplified SOA Access Control Solution

Login & Role Assignment Process is separated from typical SOA access control process (Figure 17). It is handled by a separate and independent sub-system from outside of SOA. In this process, client will accomplish login process and roles will be assigned to him/her based on his/her attributes.

Access Request & Response Process is still a part of traditional SOA structure, and it is similar to traditional access control process that we discussed in section 4.1.

Login is always the first and necessary step for all system's access. In our solution, clients or access requestors need to login first before sending access requests. Their attributes will be verified and roles will be assigned based on their verified attributes during the login process. The roles which are assigned to Access Requestors are unified within the SOA range which can be recognized and used by all individual systems.

Our design meets requirements of Single Sign On (SSO – Figure 18), which means a user logs in only once and gains access to all individual or sub systems (if she is granted necessary permission) without being prompted to login (or the process like role assignment) again at each of access request for different individual system. SSO is a major property of access control for distributed system like SOA which are composed by multiple, related, but independent software systems.

Figure 18: Single Sing on

From Figure 17 & Figure 18, we know that, in our solution, Role Assignment is not handled at access request & respond stage within SOA structure. This is different from traditional XACML or RB-XACML standards. Instead, Role Assignment and Login are both handled within one process in a separate system outside of SOA structure. The benefits for the separation are listed as following:

- Role Assignment process will not be restricted by requirement and syntax of XACML or RB-XACML standards/language. and we can implement or propose privacy protection cryptographic technology freely in a the separate Login & Role Assignment sub system to achieve privacy protection;
- Prevent personal attributes/privacy from being transported via different individual system within SOA – more privacy friendly;
- It is much easier and more efficient to maintain global Role-to-User policies in a centralized system than by individual systems;
- Role assignment and login are accomplished in one process, after that, assigned roles will be valid and usable for all access requests to all individual system. So that system efficiency is improved.

4.3.2 Login & Role Assignment Process:

Figure 19: Login & Role Assignment Process

Figure 19 provides an overview of Login & Role Assignment Process. From Figure 19, we can see that there are two independent 3rd parties, Role Assigner (RA) and Policy Verifier (PV), involved in the process. Access Requestor (AR), who is client or server Requestor in SOA, obtained eligible roles during the interactions with RA and PV during the login process. RA works as an independent 3rd party organization to maintenance Role-to-User assignment policies for all sub-systems. RA is a centralized and unified role assignment organization for SOA. RA collects and maintains individual access policies from individual systems and unifies & integrates these policies into global policies which can be applied to all individual systems.

We still have another common party which does not show up in Figure 19 – Certificate Authority (CA) which provides attribute credential for AR. The CA verifies AR's attributes and issues her credentials based on verified attributes. These attributes credentials function as private key during the role assignment process based on Identity Based Encryption (IBE) technology which we already discussed at Chapter 2. So, CA is playing a role which provides attributes verification and credential issuing for AR, and these credentials will be used as AR's private key later on during the access control process.

The Login & Role Assignment Process can be divided into the following three stages:

Stage one: AR send access request to RA, and then AR and RA will be engaged in a proposed cryptography technologies based value exchanged process. After the process AR will get a group of values which represent the attributes credentials she holds;

Stage two: AR then provides PV with the values she has after stage one, and RA also provides PV with her value and policy string;

Stage three: PV will compare the values provided by AR and policies provided by AR, then PV will assign suitable roles to AR.

During these three stages, AR & PV know nothing about AR's attributes, and on the other hand, AR and PV know nothing about RA's policies. So, it's a convinced privacy friendly process, and we will discuss the process in detail at Chapter 5.

This process has the following advantages:

- It meets with principle of Single Sign-on (SSO), that is a single login/sign on process will provide the Requestor roles for all access request within this SOA architecture;
- There is only one Role Assignment process, and roles can be used in all access requests processes within the same login session;
- Each individual system does not need to maintain Role-to-User assignment policies, and it not only greatly reduces Role Access Policies maintenance works but also provides a unified and standard way to assign roles. Individual system just needs to maintain Permission-to-Role assignment.

In conclusion, the new mode and emerging of 3rd party RA & PV make RBAC of SOA more efficient and easy to be maintained. Tedious policies integration and maintenance works are "outsourced" to 3rd party organization apart from SOA, and resource-cost "User-to-Role Assignment" process is also separated from SOA access request and is only handled once during login process based on our proposed mode. We will discuss detail Login and Role Assignment process in Chapter 5.

4.3.3 Access Request & Response Process:

Our Access Request & Response process (or we also call it Service or Information Access process) is similar to traditional SOA access control based and compatible well with XACML standard. Figure 20 illustrate general idea of our "Access Request & Response" solution.

Figure 20: Access Provider Response Process

Figure 20 is actually a simplified model of Figure 8. That means we do not need to change SOA infrastructure and current technologies. In Figure 20, PEP works as an interface with Information Requestor; PDP handles access control based on Role-permission policies (encrypted requested information by public keys of allowed roles – we will discuss the details later); PAP is the place for Permission-to-role policies; and PIP can provide PDP more requirement information (about the resources for example.) if necessary.

In our Proposed solution, roles of AR (which are granted at “Login & Role Assignment” process), work in the same way as attributes’ value in traditional SOA, and we just need a very little change to XACML standard languages. The following statement (Figure 21) is modified Request writing in XAXML on proposed RBAC model:

Modified Request:

```
<Request>
  <Subject>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
      DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name">
      <AttributeValue>sample@users.example.com</AttributeValue> </Attribute>
    <Attribute AttributeId="role" DataType="http://www.w3.org/2001/XMLSchema#string"
      Issuer="admin@users.example.com">
      <AttributeValue>Senior Developer</AttributeValue></Attribute>
    </Subject>
  <Resource>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
      DataType="http://www.w3.org/2001/XMLSchema#anyURI">
      <AttributeValue>http://server.example.com/code/docs/developer-
      guide.html</AttributeValue></Attribute>
    </Resource>
  <Action>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
      DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>read</AttributeValue></Attribute>
    </Action>
  </Request>
```

Figure 21: Sample of Modified Request for SOA RBAC Solution

Since role is regarded as attribute value of access requestor, policy language of new SOA RBAC Solution is the same as traditional XACML policy language, and response language of XACML is also the same in general. In “role-privacy” protection model, we will not use XACML response language. Instead, we will use SOAP (Simple Object Access Protocol) to transfer encrypted information to access control. SOAP is XML-based messaging framework which has the following features:

- Extensible (SOAP defines a communication framework that allows for features such as security, routing, and reliability to be added later as layered extensions);
- Interoperable (SOAP can be used over any transport protocol such as TCP, HTTP, SMTP etc.)
- Independent (SOAP allows for any programming model and is not tied to any specific system)

Since SOAP is SOA standard for message transferring, using SOAP as access response in our protocol will not break compatibility to current SOA technologies.

In conclusion, our privacy protection RBAC model does not need huge change of SOA standards which are already popularly used, and it makes our solution feasible and practical to be implemented. We will discuss detailed implementation of “Access Request & Response” process in Chapter 6.

4.4 Privacy Protection Targets for the SOA RBAC Solution

In this section, we will elaborate our targets for privacy protection. That is what privacy should be protected during the SOA RBAC process by our proposed solution?

There are two kinds of privacy which should be protected by our proposed solution: Access requestors’ (SOA clients) individual or identity privacy (including her attributes and assigned roles) and RBAC related policies (From Figure 3, we can see that there are two kinds of access control policies: Role-to-User policies and Permission-to-Role policies).

This thesis will provide solutions and protocols to protect AR’s individual privacy and all kinds of policies in both Login & Role Assignment process and Access Request & Response process. Meanwhile, we have to balance privacy protection and system feasibilities. That means we have to look into “sensitive level” of different kinds of privacy in different process stage to decide which privacy protection level action we should take. In this thesis, we will take use of the following two levels of protection action:

- ZERO knowledge disclosure. This is a strict privacy protection action. Under this action level, there will be completely no privacy to be disclosed;
- Optional and tolerant action. This action is used to protect non-sensitive privacy to gain system efficiency. Under this action, involved parties can choose to set their information as privacy (they do not want to disclose their information) or not (it is fine for them to disclose information since it is not sensitive information.). Even they set it as privacy, it may not be ZERO knowledge disclosure.

From SOA RBAC solution we proposed in section 4.3.1 and Figure 17, our access control process is composed of two parts/processes: Login & Role Assignment process; and Access Request & Response process. We will discuss these two processes in Chapter 5 and Chapter 6 accordingly. In this section, we’d like to elaborate privacy protection goals based on these two parts/processes. In generally,

- for Role Assignment process part (Figure 19), what roles AR will get depends on her attributes such as her work unit, position, location, age, income, gender etc. We regard

AR's personal attributes and RA's Attributes-to-Role policies as sensitive privacy, and we should take "ZERO knowledge disclosure" action. So,

- RA & PV will know nothing about AR's attributes after the process;
- On the other hand, AR & PV will also know nothing about RA's Role assignment policies;
- For Access Request and Response process part (Figure 20), AR will apply access permission based on her roles. We will regard the roles and SP's Permission-to-Role as non-sensitive privacy (we will provide thorough analysis in Chapter 6). As a result, we will take use "Optional and tolerant action", in which,
 - AR can set her access request as "privacy" mode or not. If it's not a privacy mode, her roles are not regarded as privacy, and her roles' information can be disclosed to SP. If it is set as a privacy mode, SP will not know AR's role information;
 - We can be tolerant on SP's Permission-to-Role policies disclosure (we will provide thorough analysis in Chapter 6).

In this Chapter, we already proposed our general RBAC solution for SOA. This solution makes perfect privacy preserving possible. Based on this solution, we will continue to work on a cryptography technologies based privacy protection solutions or protocols, and the rest parts of these thesis will illustrate our identity & policy privacy protection protocols and how they work. Our SOA RBAC solution has two processes: Login & Role Assignment process and Access Request & Response Process. In Chapter 5, we will provide protocols in privacy protection in "Login & Role Assignment process". In Chapter 6, we will discuss privacy protection in "Access Request and Response" process

5. Privacy Protection Protocols in Login & Role Assignment process

Based on our discussion in Chapter 4, our proposed SOA RBAC solution is divided into the two processes: Login and Role Assignment Process and Access Request and Response (RBAC) process. In this chapter, we will discuss privacy protection protocol in Login and Role

Assignment process only. We will talk about privacy protection protocol in Access Request and Respond process in Chapter 6.

As illustrated in Figure 19, in the login and Role Assignment process, there are three parties involved: Access Requestor (AR), Role Assigner (RA) and Policies Verifier (PV). AR is the client/consumer of SOA who sends access require for specific information or service. RA is an independent party above the composed units of SOA. RA maintains unified roles assignment policies which are agreed by all individual systems. PV is another 3rd party which gets involved in role assignment process. The purpose that we bring in PV is to achieve ZERO privacy disclosure. Policy mapping process is handled by PV as a 3rd party. In this way we can completely prevent privacy knowledge disclosure. PV's function will be discussed in section 5.2.

In this Chapter, we will focus on attributes privacy protection in Login and Role Assignment process. There are three privacy protection targets in this process:

Target One: AR's Attributes Privacy Protection: AR has to apply for roles by submitting her attributes credentials, and RA has to verify AR's credentials to decide which roles should be assigned to her. However, AR does not want to disclose her attributes because they are her privacy. However, RA has to verify AR's attributes to decide which roles should be assigned to her. Our target is to provide protocols and solutions to solve the conflict: RA can verify AR's attributes credentials and assign suitable roles to AR with **ZERO** knowledge of AR's credentials.

Target Two: RA's Policies Privacy Protection: RA & AR get involved in a credential and policies mapping process so that RA can know which roles can be granted to AR. As we stated in Target One, there is ZERO knowledge of AR's credentials be disclosed, and we also required ZERO knowledge disclosure for RA's policies. After the Role Assignment process, AR knows only which roles she is assigned and nothing else. AR has **ZERO** knowledge on RA's policies and she is not likely to guess or extrapolate RA's polices.

Target Three: All Other Privacy Information Protection: After policy mapping process, AR will be assign eligible role credentials. The role information is also AR's privacy. Our complete privacy protection action level requires no one (both RA and PV) can get to know what roles are assigned to AR.

Our proposed Login and Role Assignment process is divided into two stages: Attributes and Credentials values exchange stage: and Polices Mapping Stage.

We will discuss stage one "Attributes Credentials values exchange stage" in Section 5.1. In stage one, our major focus is on target one – AR's Attributes Credentials Privacy Protection, and we

will propose three protocols Naïve Protocol, Sophisticated Protocol and Improved Sophisticated Protocol which will be described in Section 5.1.2, 5.1.3 and 5.1.4.

In Section 5.2, our focus is stage two “Policies Mapping Stage”. In this stage, the major privacy security concern is “Target Two: RA’s Policies Privacy Protection”, and we also propose three solutions on this stage which are elaborated in section 5.2.2, 5.2.3 and 5.2.4 accordingly. In this Chapter we will also look into how our solutions realized two most important RBAC features: Role Hierarchy and Constraints.

Notations used in this chapter and the rest part of the thesis are listed in the following table.

Name	Description
$ATTR_i$	Attributes i to be verified where $i \in \{1, n\}$ and n is the total amount of attributes
k_i	The i th random number created to represent one attribute
$I(k_i, ATTR_i)$	Identity-based encryption to k_i using $ATTR_i$ as public key
$CRED_j$	Credential for attribute j where $j \in \{1, m\}$ and m is the total amount of credentials held by access requestor
$\Gamma^{-1}(I(k_i, ATTR_i), CRED_j)$	Decryption of $I(k_i, ATTR_i)$. “ $CRED_j$ ” is privacy key used for decryption.
EA	A homomorphic encryption.
DA	Decryption of “EA”
$SETINT(k_i[0], \{D_{i,1} \dots D_{i,m}\}, EA)$	A “Set Intersection”. Output is $EA[x]$, if $k_i[0] \in \{D_{i,1} \dots D_{i,m}\}$ and $x = 0$ else x will be a random number
$Enc(P, k_i[1])$	Symmetric key encryption of P using $k_i[1]$ as encryption.
OT	Oblivious Transfer
CC	Array of all values obtained by Access Requestor after value exchange stage
RCred	Credential of role
$Enc(Enc(Enc(RCred, k_1), k_2) \dots k_n)$	Symmetric key encryption to $RCred$, n times by $k_1, k_2, \dots, k_n$ as key.

$Enc^{-1}(Enc^{-1}(Enc^{-1}(RCred_s, c_n), c_{n-1}), \dots, c_1)$	Decryption “ERCred _s ” n times by $c_1, c_2, \dots, c_n$
$E_K(i)$	Identity-based Encryption on “Information” i (which is requested information) by using K as public key. K is required role (such as IT Manager) to access requested information.
$E_K(S)$	Identity-based Encryption on “S” using K as public key. S is created random number
AR	Access Requestor
RA	Role Assigner
PV	Policy Verifier
SP	Service Provider

Table 1: Notation

5.1 Attributes Credentials Values Exchange Stage

Attributes Credentials values exchange stage is stage one of Role Assignment process. In this stage, RA will verify AR’s credentials and result of verification will be used at stage two as necessary input for policies mapping, and the verification process is privacy preserving.

5.1.1 Attributes Standards and Preparation

Our protocol is based on Identity-Based Encryption (IBE) technology discussed in 3.1. Attributes will be used as Identity Key, and we need to frame attribute standard to facilitate the use of IBE.

In our protocol, attributes are all identical and clear strings. For example, for “Age” attributes, we used to use number to represent person’s age such as 28, and range of numbers to represent a policy such as “age between 20 and 30” or “>= 20 and <= 30”. We can not use this way to define attributes and policies, because number attributes which are not identical strings, and can not be used in IBE.

Instead, we define attributes and policies in this way. For the same sample of age, it can be represented by the following identical strings: “Age below 20”; or “Age between 20 and 30”; or “Age between 30 and 40”; or “Age above 40” etc.

So, RA can also use one of the string such as “Age between 20 and 30” in her policies pool which can be used as public key during the value exchange process in our protocol. On the other hand,

AR may hold a credential as *CRED* issued by a 3rd party Certificate Authority or Public Key Generator (PKG) for “Age between 20 and 30” which will be used as private key. As we discuss in section 3.1, this is realization of Identity-Based Encryption (IBE). RA holds clear strings of attributes in her policies which will be used as public keys, and AR holds a bunch of credentials issued by 3rd CA or PKG which will be used as private key. This is implemented based on principle and requirement of IBE.

For example, RA has a policy to assign role of “Senior Developer” which is a combination of the following attributes:

Attribute One: *In a core sub-organization* (individual system/organization can be divided into several levels based on their business, size, importance etc.);

Attribute Two: *Work Experience more than 5 years;*

Attribute Three: *Position in Software development;*

Attribute Four: *Salary between 60k and 80k;*

So, with IBE technology, during the trust negotiation process in login, RA can use each of attributes in policy such as “Position in Software development” as public key to encrypt values to be exchanged in our proposed protocol. AR obtains her private keys for verified attributes from CA or PKG as credentials. Only when AR holds all credentials for required attributes, she will be granted the role. More details about the technology can be found in [6]. We will discuss how RA and AR use attribute / credentials as public key / private key to exchange values as trust negotiation in this chapter

5.1.2 Privacy Protocol – I: Naïve Protocol

For this stage, we propose three protocols: Naïve Protocol, Sophisticated Protocol and Improved Sophisticated Protocol.

In Naïve Protocol (Figure 22), the process & principle is simple. The RA creates one random value for each attributes in all her policies, and encrypts the value by corresponding attribute. Then RA sends these encrypted values to AR. After receives the encrypted values AR will try to decrypt the value by her credentials (private key). For each decryption operation, AR will get a decrypted value. But AR has no idea if the credential (private key) matches required attributes. AR has to try to decrypt each value by all her credentials. After the process, AR will for sure get correct value for some of attributes for which she holds corresponding credentials. However, she

totally has no idea how many attributes and what attributes she matches. There are only two steps are involved:

Figure 22: Naïve Protocol

Step 1: For each attributes $ATTR_i$ that may relate to any one role's assignment (for example, there are total 20 roles, and total $n = 200$ attributes are involved, $i \in \{1, 200\}$), RA creates one random key k_i for each $ATTR_i$, and she encrypts k_i by $ATTR_i$, $E_i = I(k_i, ATTR_i)$. Then, RA send E_i $i \in \{1, 200\}$ to AR;

Step 2: AR receives α_i $i \in \{1, 200\}$, for each α_i , she decrypts it by each credentials ($CRED_j$) she holds. $D_{i,j} = I^{-1}(E_i, CRED_j)$. If AR totally have m credentials, then AR will get m values $D_{i,1} \dots D_{i,m}$ for each E_i . If one of $CRED_j$ match $ATTR_i$, then AR will get the value of k_i , or she will get a random number.

In conclusion, for each attributes,

$ATTR_1 \rightarrow$ AR creates k_1 and encrypts it to $E_1 \rightarrow$ RA gets a group decrypted values of $D_{1,1} \dots D_{1,m}$

$ATTR_2 \rightarrow$ AR creates k_2 and encrypts it to $E_2 \rightarrow$ RA gets a group decrypted values of $D_{2,1} \dots D_{2,m}$

...

$ATTR_n \rightarrow$ AR creates k_n and encrypts it to $E_n \rightarrow$ RA gets a group decrypted values of $D_{n,1} \dots D_{n,m}$

Only when RA holds a right credential, there will be a β value matches corresponding k . However, both RA & AR do not know which values are matched and how many values are matched. In other word, RA knows nothing about AR's credentials. On the other hand AR does not know matched values neither, it's not possible for AR to deduce RA's policies. This is prerequisite of policies protection. We will discuss it in details in next section.

Now, let's analyze the result. If AR has $m = 20$ credentials ($CRED_j, j \in \{1, 20\}$), giving $n=200$ (RA has 200 attributes totally). AR will totally have values of:

$$m * n = 20 * 200 = 4,000.$$

If among AR's total 20 credentials, there are 15 credentials match attributes required by RA, there will 15 correct k values within AR's 4,000 values. Again, both AR and RA do not know which 15 values are matched.

During the process, AR has no idea about how many attributes she matches, and what attributes she matches. So it's not feasible for AR to deduce Role-Attributes policies. On the other hand, RA knows nothing about AR's credentials because "Step 2" happens at AR side and AR does not need to send anything to RA. So, this protocol meets privacy protection requirement.

After the process is finished, AR will pass all $m * n$ values to PV, and RA will also pass $k_i \in \{1, n\}$ and policies (PV will not understand the policy because it's represented by k values). We will discuss the policy verification process later on in this chapter.

Naïve Protocol is not feasible because it will cause values explosion when both m (amount of AR's credential) and n (amount of RA's attributes) values are very large, and it is always the case in a true distributed system in industry. However, Naïve Protocol is easy to be understood. So, it can be helpful for us to understand the Sophisticated Protocol that we will discuss in next section.

5.1.3 Privacy Protocol - II: Sophisticated Protocol

Naïve protocol looks simple, however, we can easily find its shortcoming: not scalable. For example, if $n = 2000$ and $m=500$, AR will totally provide $m * n = 500 * 2000 = 1,000,000$ values to PV. The number is still possible to increase quickly with growing of the system. The gigantic amount of values will cause decrease of system in several ways such as it needs more time and bandwidth to transfer the values from AR to PV; and it needs more time of policy mapping in PV.

So, we propose another protocol: Sophisticated Protocol which is more complicated in cryptography point of view, however, AR will only provide n values to PV for further process. In sophisticated protocol, if AR holds only 15 valid credentials for attributes, she will get only 15 valid values after value exchange process with RA. As a result, she just needs to pass these 15 values to PV for evaluation. So, we solve the problems of "scalability" and "efficiency".

Figure 23: Sophisticated Protocol

From Figure 23 we know that sophisticated protocol contains 5 steps. We will explain the protocol and these steps in details in this section.

Step 1: For each attributes ($ATTR_i$) required by all roles, RA created two random keys $ki[0]$ & $ki[1]$, and a public marker P . RA encrypts each $k_i[0]$ by $ATTR_i$: $E_i = I(K_i[0], ATTR_i)$. Then, she send all E_i $i \in \{1, n\}$ (RA totally has n attributes) and P to AR. This step is pretty like the step 1 in Naïve Protocol expects that RA create TWO keys for each attributes rather than only one.

After this step, for each attribute $ATTR_i$, RA will have the following values:

- Random key pair ($ki[0]$, $ki[1]$);
- A public marker P ;
- Encrypted $E_i = I(K_i[0], ATTR_i)$

Among them, all E_i and P will be sent to AR for further process.

Step 2: AR then creates a semantically secure homomorphic encryption system EA . Then for each value α_i , and for each of her credentials $CRED_j$, she creates a value $D_{i,j} = I^{-1}(E_i, CRED_j)$. If

she holds m credentials, then she will get m D values for each E_i ($D_{i,1} \dots D_{i,m}$). Comparing “Step 2” in naïve protocol, AR need to create a semantically secure homomorphic encryption system EA , and the other parts are the same.

After this step, AR will create homomorphic encryption system EA ; and for each received E_i , get a group values of ($D_{i,1} \dots D_{i,m}$) which are created by $D_{i,j} = \Gamma^{-1}(E_i, CRED_j)$ for every credential she holds;

Step 3: AR and RA then engaged in Set Intersection protocol, $SETINT(k_i[0], \{D_{i,1} \dots D_{i,m}\}, EA)$. AR’s input are $\{D_{i,1} \dots D_{i,m}\}$ and EA , and RA’s input is $k_i[0]$. As a result, if there is only one value in $\{D_{i,1} \dots D_{i,m}\}$ equal to $k_i[0]$ (AR has the credential of required attribute), RA will learn $EA[0]$, otherwise it will be a random value.

So, after this step, for each attributes (or each E_i she sent to AR), RA will get the value of

$$EA[x_i] \text{ (} x_i = 0, \text{ if AR has the credential, or } x_i \text{ will be a random number.)}$$

Step 4: RA calculates

$$\delta_i = EA[x_i] * EA[k_i[1]] = EA[x_i + k_i[1]]$$

Then, she makes ordered pairs $(\delta_i, Enc(P, k_i[1]))$. The order of pairs is random. RA then sends these pairs to AR.

After this step, for each attribute or value of she creates the following two values and sends them to AR:

$$\delta_i = EA[x_i] * EA[k_i[1]] = EA[x_i + k_i[1]]$$

$$\& Enc(P, k_i[1])$$

Step 5: When AR receives the pairs, she computes $\eta_j = DA(\delta_j)$ and then computes $Dec(Enc(P, k_j[1]), \eta_j)$. If AR has the credential of $ATTR_j$, $\eta_j = k_j[1]$, and she will get P or she will get a random number. If AR get P , she will keep $k_j[1]$ which is a valid value to prove she holds a valid credential (even though she does not know which one she holds).

After the step, AR will get P when she could successfully decrypt E_i by $D_{i,j} = \Gamma^{-1}(E_i, CRED_j)$ at Step2 and let $EA[x_i]$ ($x_i = 0$) at Step 3. Or AR won’t know P .

Our protocol may stop here. If AR totally has 15 credentials match attributes for example, she will have 15 $k_j[1]$ values which will be sent to PV for further process. However, AR does not

know what attribute or credential these values represent. She just knows she have 15 valid values which means she has 15 valid credential which may help her to be granted a specific role.

So, AR knows nothing about RA's policy, and on the other hand, RA knows nothing about AR's credentials. However, AR will at least get to know one thing: how many valid credentials she holds. We can improve our protocol so that AR does not even know how many valid credentials she holds. We call it Improved Sophisticated Protocol, and discuss it at next section.

5.1.4 Privacy Protocol - III: Improved Sophisticated Protocol

Improved Sophisticated Protocol is proposed for enhanced privacy preserving. Under this protocol, AR will not even know how many valid credentials she holds. Of course, AR still knows nothing about RA's policy and RA knows nothing about AR's attributes.

Figure 24: Improved Sophisticated Protocol

Step 2 to Step 3 of this improved protocol are the exactly the same as Sophisticated Protocol we discussed at previous section. We will modify Step1, Step 4 and Step5 in this new protocol and add one more step – Step 6. Now let's view details:

Step 1: For each attributes ($ATTR_i$) required by all roles, RA created two random keys $k_i[0]$ & $k_i[1]$. RA encrypts each $k_i[0]$ by $ATTR_i$: $E_i = I(K_i[0], ATTR_i)$. Then, she send all E_i $i \in \{1, n\}$ (RA totally has n attributes) to AR.

We can see, Step 1 is pretty like Step 1 in Sophisticated Protocol. Only difference is that RA does not create P value as public marker.

Step 2: AR then creates a semantically secure homomorphic encryption system EA . Then for each value α_i , and for each of her credentials $CRED_j$, she creates a value $D_{i,j} = I^{-1}(E_i, CRED_j)$. If she holds m credentials, then she will get m D values for each E_i ($D_{i,1} \dots D_{i,m}$). Comparing “Step 2” in naïve protocol, AR need to create a semantically secure homomorphic encryption system EA , and the other parts are the same.

After this step, AR will create homomorphic encryption system EA ; and for each received E_i , get a group values of ($D_{i,1} \dots D_{i,m}$) which are created by $D_{i,j} = I^{-1}(E_i, CRED_j)$ for every credential she holds;

Step 3: AR and RA then engaged in Set Intersection protocol, $SETINT(k_i[0], \{D_{i,1} \dots D_{i,m}\}, EA)$. AR's input are $\{D_{i,1} \dots D_{i,m}\}$ and EA , and RA's input is $k_i[0]$. As a result, if there is only one value in $\{D_{i,1} \dots D_{i,m}\}$ equal to $k_i[0]$ (AR has the credential of required attribute), RA will learn $EA[0]$, otherwise it will be a random value.

So, after this step, for each attributes (or each E_i she sent to AR), RA will get the value of

$$EA[x_i] \text{ (} x_i = 0, \text{ if AR has the credential, or } x_i \text{ will be a random number.)}$$

Step4: RA calculates

$$\delta_i = EA[x_i] * EA[k_i[1]] = EA[x_i + k_i[1]]$$

Then, she sends δ_i to AR. Compared with Sophisticated Protocol, RA does not create $Enc(P, k_i[1])$, and just creates and sends the following one value to AR for each attribute:

$$\delta_i = EA[x_i] * EA[k_i[1]] = EA[x_i + k_i[1]]$$

Step5: When AR receives the pairs, she computers $\eta_j = DA(\delta_j)$. Since there is no public marker P , she will not know if the value she get by $DA(\delta_j)$ is $k_i[1]$ or not.

After the step, for each required attributes, AR will get a value $k_i[1]$ (if she holds the credential for the attribute) or a random value (if she does not holds the credential). But, AR totally does not know if she gets a $k_i[1]$ or a random number.

This protocol can stop here because AR can provide PV enough information to handle role assignment. However, we can still add one more step as optional:

Step6: For each attribute $ATTR_i$ (with the same order of pairs in step 4), RA and AR engaged in a 1-out-of-2 OT protocol. Where RA's input is the list of $\{r_i[0], \text{Enc}(r_i[1], k_i[1])\}$, and AR's input is η which is actually $k_i[1]$ if she has it or a random value if she does not have the credential. If her input is $k_i[1]$, she will decrypt $\text{Enc}(r_i[1], k_i[1])$, and gets the value of $r_i[1]$ or she will get $r_i[0]$.

In this case, AR will send list of $r_i[1]$ or $r_i[0]$ $i \in \{1, n\}$ to PV. For example, if AR has 15 valid credentials, she will have 15 $r_i[1]$ and the others are $r_i[0]$. After this step, AR will not hold a random number, her values provided to PV for evaluation will be one of value of $\{r_i[0], r_i[1]\}$. This will be same as $\{0, 1\}$ or $\{\text{no}, \text{yes}\}$ for PV if PV get the attributes-to-role policy from RA. Our protocol makes this step as optional because it's not necessary.

5.1.5 Comparison of Three Protocols

Table 1 provides comparison of three proposed protocols in aspects of security, involved technologies and system efficiency (steps, calculations and operations and created values etc.).

	Naïve Protocol	Sophisticated Protocol	Improved Sophisticated Protocol
Cryptographic Technologies	<ul style="list-style-type: none"> Identity-based Encryption 	<ul style="list-style-type: none"> Identity-based Encryption Homomorphic Encryption Set Intersection 	<ul style="list-style-type: none"> Identity-based Encryption Homomorphic Encryption Set Intersection 1-out-of-2 OT
Steps	2 steps	5 steps	6 steps
Calculations & Operations	n: encryptions m*n: decryptions	2n: encryptions m*n + 2n: decryptions n: Set Intersection protocol operations	2n: encryptions m*n + 2n: decryptions n: Set Intersection protocol operations

		n: Homomorphic Encryption operations	n: Homomorphic Encryption operations n: 1-out-of-2 OT operations
Created Values	$m \cdot n$	valid credentials number (≥ 0 & $\leq m$)	n
Security	Zero Knowledge Disclosure	AR know valid credentials number	Zero Knowledge Disclosure
“n” is total amount of attributes involved in RA’s policies; “m” is total amount of credentials held by AR			

Table 2: Comparison of Three Protocol for Value Exchange Stage

5.2 Policies Mapping Stage

In 5.1 we discussed several protocols to let RA verify valid credentials held by AR in a totally blind way. After credential verification process, AR will obtain a group of values among them some values represent valid credentials she holds. In RA side, she has role assignment policies (attributes-to-role), for example, a policy is to assign a “Senior IT Manager” role, and it requires three attributes: 1. Employer: First level corporation; 2. Position: IT Manager; 3. Duration in the position: 5 years and more

These three attributes will be changed to three values in RA’s policy, such as (k_1, k_5, k_{10}) . Then, the next step for Login and Role Assignment process is to check if AR holds these three values or not. If yes, AR will be assigned the role, or she will not be assigned the role. This step is still handled by a privacy-preserving method, and we will discuss the method in details in this section.

5.2.1 Principles for Policies Mapping and Role Assignment

Let’s define input of RA and AR during the stage.

After values exchange stage using proposed protocol discussed in 5.1, AR has a bunch of values, and some of these values represent credentials she holds. Let's briefly review the values held by AR:

- In Naïve Protocol: AR has n (RA's attributes amount) rows of values, and each role has m (AR's credentials amount) values. So, if AR has 15 valid credentials, she will get 15 valid values ($m \geq 15$), and each row can have no more than one valid value;
- In Sophisticated Protocol: if AR has 15 valid credentials, she will have 15 valid values (each value will represent one valid credential);
- In Improved Sophisticated Protocol: AR will have n (amount of all attributes involved in RA's policies) values $v_i, i \in \{1, n\}$, and $v_i \in \{r_i[0], r_i[1]\}$. If AR holds a valid credential according to a required attribute $ATTR_i$, $v_i = r_i[1]$ or $v_i = r_i[0]$;

These values will be mapping to RA's policies which are also represented by a group of values and Roles will be assigned to AR based on values mapping result.

RA has a bunch of policies, and each policy decides if a Role should be assigned to AR or not. We use function $p(k_1, k_2 \dots k_h)$ to denote a policy. $\{k_1, k_2 \dots k_h\}$ are values created by RA during the process we discussed in 5.1. If AR owns a required credential, she will have a related k value. Let's form a value set of K , and set a function of $P\{K\} = p(k_1, k_2 \dots k_h)$, when $k_1, k_2 \dots k_h \in K$, $P\{K\} = p(k_1, k_2 \dots k_h) = 1$.

Or, we can define the principle in this way:

RA: RA has a policy $p : \{0, 1\}^h \rightarrow \{0, 1\}$, and corresponding h key values $\{k_1, k_2 \dots k_h\}$. If one key is matched from values held by AR, the value $\{0, 1\}$ will be set to 1, and only all values matched, the value of the function will be $\{1\}$, that is $p : \{1\}^h \rightarrow \{1\}$;

AR: AR can provide a bunch of values she obtains during the process described in 5.1. In Naïve protocol case, she will has $m * n$ values; in sophisticated protocol cased she will have x ($x \leq m$ & $x \leq n$) values which means she holds x valid credentials; in improved sophisticated protocol, she will get n values. Let's use $CC = \{c_1, c_2, \dots, c_{m*n}\}$ (in Naïve protocol), $\{c_1, c_2, \dots, c_x\}$ (in Sophisticated protocol) and $\{c_1, c_2, \dots, c_n\}$ (in Improved Sophisticated protocol) denote the value set AR has.

Mapping Principle: If there is a subset of $C_x \in CC$, and $C_x = \{k_1, k_2 \dots k_h\}$ (which is value set of a policy), $P\{C_x\} = 1$, that is $P\{C_x\} = 1$ when AR (value set of CC) has all required key ($\{k_1, k_2 \dots k_h\}$) in RA's policy.

What if Matched: If AR's value set match RA's policy, AR will receive a credential of related Role, which can be used as private key of Role IBE during the information access stage (we will discuss it in next chapter).

Multiple Polices for One Role: It's possible for multiple policies defined for the same Role, and AR can match more than one policy for just one Role. In that case, we can let algorithm to reduce duplicates, and AR will get one Role in this case, and all Roles AR gets are identical.

How to handle policies mapping: How to handle policies mapping is a tricky process. It relates privacy and computation complication issues. We will suggest 3 solutions for Policy Mapping. In Solution One and Solution Two, Policy Mapping is handled directly between AR & RA. In Solution Three, a 3rd party is involved in the process.

Role Hierarchy and Constraints: Role Hierarchy and Constraints are important features in RBAC, and we will discuss them in 5.2.5 on how to handle Role Hierarchy and Constraints during role assignment process.

5.2.2 Solution One for Policy Mapping

Solution One: RA handles Policy Mapping

Figure 25: Policy Mapping - Solution One

In this solution illustrated in Figure 25, AR will send her values directly to RA. After receiving values from AR, RA will compare and map these values with required keys in her policies to decide which Roles should be assigned to AR. Then, RA sends matched roles' credentials (we denote them in $RCred_s$) to AR.

Obliviously, this solution violates privacy protection principle in two ways. First since the key values are provided by RA herself, she will get to know which value represent what attributes. For example, if she checks a value of k_i , and she knows this key is for attributes of an “Age between 20 and 30”, then AR’s privacy is disclosed. Actually, this is almost the same as AR passes her credentials directly to RA. Second, RA will get to know which Roles the AR will get.

Solution One does not meet our privacy preserving requirements. Actually, if we go with this solution, all protocols we discussed in Section 5.1 are meaningless.

5.2.3 Solution Two for Policy Mapping

Solution Two: Cryptographic Solution for Policy Mapping

If AR matches all attributes (she has all related key values), she will be assigned credential of related Role ($RCred_s$). In RA’s policy, there are n attributes required for this Role, and there are n key values correspondingly $\{k_1, k_2 \dots k_n\}$. RA must make sure that AR owns all these keys to get $RCred_s$. Figure 26 provide us with steps of solution two for Policy Mapping.

Figure 26: Policy Mapping - Solution Two

Step 1: RA picks up n random numbers $\{q_1, q_2, \dots, q_n\}$. Then she encrypts these n numbers by using $\{k_1, k_2 \dots k_n\}$ as key (Note: the order is fixed, for example, k_1 is used to encrypted q_1 , and k_n is used to encrypt q_n .) $Eq_j = \text{Enc}(q_j, k_j); j \in \{1, n\}$. So, RA get n encrypted values $\{Eq_1, Eq_2, \dots, Eq_n\}$.

Step 2: RA encrypted $RCred_s$ values, n times by $\{k_1, k_2 \dots k_n\}$ in the same order of she created Eq_j values. RA will get:

$$ERCred_s = \text{Enc}(\text{Enc}(\text{Enc}(RCred_s, k_1), k_2) \dots k_n);$$

Then RA will pass the $\{q_1, q_2, \dots, q_n\}$, $\{Eq_1, Eq_2, \dots, Eq_n\}$ and $ERCred_s$ to AR.

Step 3: After AR receives these values, she will decrypt Eq_n first by each of key values she gets from process described in 5.1. $Y = \text{Enc}^{-1}(Eq_n, c)$. $c \in CC$. If there is one Y value equal to q_n , then the c is k_n , and AR will keep the value and try to decrypt the other value of $Eq_{n-1} \dots$ until Eq_1 . If AR can successfully decrypt all values in $\{Eq_1, Eq_2, \dots, Eq_n\}$, she will get all required key values $\{c_1, c_2 \dots c_n\}$ which is equal to $\{k_1, k_2 \dots k_n\}$.

Step 4: Then she will use the keys to decrypt $ERCred_s$ by $k_n, \dots, k_2, k_1$. (in the fixed order), and she will get $RCred_s$ as Role's credential:

$$RCred_s = \text{Enc}^{-1}(\text{Enc}^{-1}(\text{Enc}^{-1}(ERCred_s, k_n), k_{n-1}) \dots k_1);$$

If AR is failed to get any one value in $\{q_1, q_2, \dots, q_n\}$, that means she can not have all keys in $\{k_1, k_2 \dots k_n\}$. She is not qualified for the Role's credential. AR will just stop the process.

Is Solution Two a complete privacy preserving solution? After this process, AR will get to know the following information:

- How many valid attributes credentials he has (amount of valid c values);
- How many valid credentials RA requires for each role;
- Which values are required for a specific role

So, even though RA doesn't know anything about AR's credentials, it's not a complete privacy preserving solution.

In addition, it will cause computation and complication issue. Let's use naïve protocol as a sample, the AR will have 1,000,000 values, and if the Role need 50 attributes ($n = 50$), then AR will have to do up to $1,000,000 * 50 = 50,000,000$ times decryption to get the *RCred_i*.

So, we will suggest another solution.

5.2.4 Solution Three for Policies Mapping

Solution Three 3rd party PV for Policies Mapping

Model displayed by Figure 19 is using the solution. To solve privacy and complication issues together, the 3rd party PV will handle Policies Mapping task.

PV will receive all policies list from RA ($p_1(k_{11}, k_{12}, \dots, k_{1n1}), p_2(k_{21}, k_{22}, \dots, k_{2n2}), \dots$), and all AR's key values set CC., then PV can map policies with AR's value by the following operation (Figure 27):

1. PV pickup one policy value set (received from RA);
2. Check if the Role is constraint with Roles already assigned to AR, if yes, error or warning, if no, continue (details in section 5.2.5);
3. Check if its parent role has already assigned to AR, if yes, assign this role to AR, if no, continue (details in section 5.2.5);
4. Pickup one value in policy value set, and search the value from AR's value set. If matched value is found, then pickup the next value in policy value set, if not found, return to step 1 to pickup another role's value set;
5. Repeat step 4 until the last value;
6. If all values are matched (end of value set), then assign the role to AR;
7. Repeat step 1 until the last policy value set (end of policy value set);

Figure 27: Flow Chart for Policy-Role Mapping Process

The flow chart or algorithm can be implemented by any suitable programming language based on PV's platform and system. In this thesis, we will not provide source code for the operation.

Now, let's verify privacy protection for this stage:

PV receives all policies which are represented by sets of values from RA. PV has no knowledge of which value represents which attribute, as a result, PV knows nothing about RA's policies.

With the same reason, PV knows nothing about AR's credentials neither because PV has no idea about relationship between the value and represented credentials/attributes.

We can safely conclude that bringing in 3rd party PV can perfectly protect privacy during role mapping process.

5.2.5 Comparison of Three Solutions

Table 2 compares three proposed solutions in security, technologies, complexity and system efficiency points of view.

	Solution One	Solution Two	Solution Three
Mapping Location	RA	RA for encryption AR for decryption	PV
Technologies	Mapping	Cryptography-based	Mapping at PV
PV involved?	No	No	Yes
Steps	N/P	4 steps	N/P
Calculation and Operations	Policy-role mapping	$2*s*1$ encryptions up to $2*v*s*1$ decryptions	Policy-role mapping

Security	Not Privacy Preserving	AR will know: <ul style="list-style-type: none"> • How many valid attributes credentials he has; • How many valid credentials RA requires for each role; • Which values are required for a specific role 	Zero Knowledge Disclosure
<p>“s” is total amount of RA’s role credentials; “l” is average amount of required attributes for each role credential in RA’s policies; “v” is created values held by AR after value exchange stage</p>			

Table 3: Comparison of Three Solutions for Policies Mapping Stage

5.2.6 Role Hierarchy and Constraints

As we discussed in section 1.3, Role Hierarchy and Constraints are two important features in RBAC model. To handle Role Hierarchy and Constraints in our solutions, we propose the following principles:

Principle One: The most senior role first

When RA provides policy value sets to PV, the order of the policy value sets is arranged. The most senior role’s value sets will be verified first. Let’s see the sample presented in Figure 1.4. The role of “IT Manager” will be first verified, then “Senior Developer” and the “Junior Developer” is the last role to be verified.

Principle Two: Children Roles Follows Parent Role

When a senior role (parent of children roles) is assigned to AR, all children role under this role will also be assigned to AR without value verification. In sample of Figure 1.4, if role of “IT Manager” (which is verified first) is assigned to AR, roles of “Senior Developer” and “Junior Developer” which are children role of “IT Manager” will also be assigned to AR without value verification process.

Principle Three: Zero Tolerance to Violation of User-to-Role Assignment Constraint

In section 1.3, we talked about role assignment constraint issue. Some of conflict roles can not be assigned to the same user. For example, role of “Senior Developer” and “Senior QA” can not be assigned to the same person.

In our solution, we are zero tolerant to violation of such constraint due to security consideration. When conflict roles are assigned to the same person, PV will stop Policy mapping and value verification process, and send error message to both AR and RA. No role will be assign to AR. Both AR and RA will review their attributes and policies to find out where problem is and fix it. For example, if a both “Senior Developer” and “Senior QA” roles are assigned to AR, then either there is something wrong for AR’s attributes credentials (she can not hold credentials which will match both “Senior Developer” and “Senior QA” role, or there is some problem for RA’s policies (the policies should be well defined that a normal user can match policies of these conflict roles). After the problem is found and fixed, they will restart a new login and role assignment process.

5.2.7 Role Credential and Role Assignment

Matched role(s) will be assigned to AR after policies mapping and values verification process. During process of solutions three we defined in 5.2.4, matched roles are recorded, so PV knows which roles should be assigned to AR even though he does not know which roles’ title they are.

We talked about Identity-Based Encryption (IBE) technology in Section 3.1. We already use IBE technology in three values exchange protocols we proposed in 5.1, and we will continue to use the technology for information access process. When a role is assigned to AR, actually, it’s the credential of the role is assigned. AR holds the role’s credential and uses it as private key. On the other hand, service/information provider uses the string to role’s title as public key. We will discuss its details in Chapter 6.

Let’s go back to Policy Mapping Solution Three. During the initiation stage in the flow chart, RA will send all policies value sets to PV as well as all role’s credentials (again, these credentials will be used as private key in IBE protocol.). These credentials are encrypted by a secret key S shared by AR & RA. The secret key S can be easily created during the vales exchange protocols between AR & RA. So, PV will receive $Enc(RCred_s, S)$, and PV still knows nothing about Role credentials. After policies mapping process, PV will send matched encrypted Role Credentials to AR. AR already has shared key (in key exchange stage with RA) to decrypt Role Credential by $Enc^{-1}(RCred_s, S)$ to get role credentials $RCred_s$.

Then, AR has finished all steps to get her Role Credential without disclosing any of her attributes, and RA also successfully keeps her policies in secret.

5.3 Security Review for Role Assignment Process

Let's take a brief review on our protocols and solutions proposed in stage one and stage two for Role Assignment Process to see if it meets our two privacy protection targets or not:

Target One: AR's Attributes Privacy Protection – ZERO disclosure of AR's attributes;

Target Two: RA's Policies Privacy Protection – ZERO disclosure of RA's Policies

Target Three: All Other Privacy Information Protection – such as RA's role's credentials and AR's roles information;

In Stage one, we proposed three protocols with different complication level. In these three protocols, RA encrypted values by attributes as public key using IBE, and there is no non-neglectful possibility for AR to decrypt these values without credentials as private key. Or IBE is unsustainable. So, AR can not get to know these values if she does not hold valid credentials. This is the security base for these three protocols.

In these three protocols, encrypted values are passed from RA to AR to handle. So, RA receives nothing about AR's credential and as a result, RA of course has ZERO knowledge about AR's credentials/attributes.

Since attributes in RA's policies are represented by random values in arbitrary order, In AR's side, during her process of decrypting arbitrary ordered encrypted random values, AR is not possible to infer which value represents which attribute. In Sophisticated Protocol, AR will get to know how many valid credential she has, if this is a privacy security concern, we can use Naïve Protocol and Improved Sophisticated Protocol which are complete ZERO knowledge disclosure protocols. AR even does not know which values are matched which are not among the bunch values she gets. So, AR will also have ZERO knowledge about RA's policies.

In Stage two, we created three solutions. Among them, solution one and solutions two are proved to be not privacy-preserving. We still put Solution One & Solution Two in this chapter because it can be used as a reason why we need to use a third party PV in Solution Three. Now, we just analyze security in Solution Three.

First of all, in Solution Three, there is no direct information/value exchange between AR & RA, so there is of course ZERO privacy disclosure for both AR & RA.

Then, let's see what PV can know from AR & RA.

PV receives policies values and encrypted Role Credentials from RA. PV can not decrypt Role Credentials without shared key between AR, and policies values are also meaningless to PV. So, PV has ZERO knowledge to RA's policy and role credentials.

AR sends PV a bunch of values getting in Stage One, which are also meaningless to PV. So, PV has ZERO knowledge about AR's attributes. After policy match, PV will send matched encrypted Role Credentials to AR. PV does not know what roles AR will get, but PV does know how many roles AR will be assigned. If this is a privacy concern, it can be easily fixed by the following way:

RA can create some fake polices with fake Role Credentials to let true Role Credentials hidden in these fake policies. Then after Policy Mapping process, PV will get to know how many Roles are matched, but there are some fake roles included. As a result PV does not know how many true/valid roles AR has. That's ZERO knowledge of AR's roles.

Now, we can safely conclude that our protocols and solutions are complete privacy-preserving with ZERO knowledge disclosed.

6. Privacy protection in Access Request & Response Process

6.1 Goals and Principle for Our Privacy Protection Protocols

Our protocol proposed in Chapter 5 meets with SSO (Single Sign-on) feature of SOA. AR just needs to sign on/login only one time, and she will be able to apply for accessing all web services or information within the SOA structure without login again.

In this chapter, we will look into privacy protection in Service or Information access process (or we can also call it as Access Request & Response process). For AR, she has already obtained a bunch of Role Credentials after Login and Role Assignment process. These role credentials will be used to decide if access permission should be granted to her or not. Normally, AR's role information is not regarded as sensitive personal privacy. However, in some cases, AR still would not like service/information providers know her roles. For example, a VP does not want other person know he is checking work process or reviewing some information. So, in this process, for

this kind of cases AR regards her assigned roles credentials as privacy, and she needs an option to set her roles credentials as privacy.

For SP's role-to-permission policies which provide permission granting rules, there is a trade-off between privacy and performance. First, we need to analyze importance of role-to-permission policies protection in this process comparing with attributes-to-role policies protection in Login & Role Assignment process. Attributes-to-role policy is privacy sensitive because if AR gets to know the policies she will try to forge attributes (even though attributes credential is issued by a 3rd party CA, attribute verification is a highly customized process and fake attributes could still be able to pass verification process if AR has intention to do so.) Since we already have ZERO knowledge disclosure protocols discussed in Chapter 5 and no AR can get to know attributes-to-roles policies. So, even AR knows the policies and she can not cheat to get that role's credentials because she can not know attributes-to-roles policies. AR can not obtain a role credential if she really holds necessary attribute. Under this situation, roles-to-permission policies are not privacy sensitive any more. In addition, there is a common sense that more senior roles can have more permission. For example, if AR's role is "Junior Developer" and fails to get permission to edit a code, she will easily assume that the permission should be assigned to "Senior Developer". Subjectively, AR always wants the most senior roles to have more access permission no matter she knows (or reason out) roles-to-permission policies or not. As a result, protecting roles-to-permission is not critical in our privacy preserving goal.

From performance and implement point of view, a complete privacy protection protocol (ZERO privacy/knowledge disclosure for both sides) is very complicated and resource consuming just like protocols we proposed in Chapter 5. It's acceptable for login and role assignment process, because: first, it's a once for all process. AR just needs to login once and she accesses information many times without login again (Single Sign On) within the same session. Second, it's a separate system from SOA, and we can implement our protocols by whatever technologies without worrying about limitation of SOA standards. We can imagine if a similar protocol is implemented for service or information access process, we need to repeat the whole cryptographic calculations for each access request, and the implementation must be compatible with current SOA technologies and standards. That will cause huge performance lose and implementation difficulties.

Combining the critical level of privacy and impact on system performance, we can safely conclude that in Access Request & Response process, policies (roles-to-permission) is not our

major goal of privacy protection protocols, and we will take “Option and Tolerant Action” in this process.

Based on nature of Access Request & Response process, in some cases, SP has to know the result of access request – success or fail. For example, if the access request is to visit a webpage provided by SP, SP will finally for sure to get to know if AR is able to access or not. SP will also know if AR has required roles or not. Our privacy protection can not prevent this from happening.

Based on above analysis, we can list our goals and principle for privacy protection in service & information access process:

One: AP can choose to set roles as privacy or not when she sends out access request. If roles are set as privacy, SP will have ZERO knowledge about AP’s roles except for the case that AR needs to visit SP’s services such as a webpage. In that case SP will know the result of access request;

Two: We will also try to keep SP’s roles-to-permission policies as secret, however, based on our analysis, the policies are not privacy sensitive. So, it is not the major goal for our protocols.

Three: service and information access has to be based on SOA technologies such as web service, SOAP, XML, SOA RBAC profile and standards such as XACML and SAML etc. So, our solution should be compatible or be able to be implemented with current SOA technologies and standards.

6.2 Our Role Privacy Protection Protocol

Role’s Privacy may not be as sensitive as personal attributes such as company, position, age and gender etc., however, it’s still desirable and valuable in some cases for Role’s Privacy protection. For example, a top manager wants to access some information as inspecting and supervising, and he does not want the SP know a top manager is checking the data/information. At this scenario, Role Privacy protection technology is required. So, in most of cases, roles information is not regarded as privacy, for example, role of “IT Developer” is held by thousands of developers in a large organization, and it’s not possible for SP to know who is accessing the information. However, role of “Vice President of IT” is held by very limited person, and it’s easily for SP to reason out who is accessing the information. In that case, AR is likely to require role privacy protection during the access process.

As we mentioned in section 6.1, there are two kinds of access request, “service” (such as a webpage) access request or “information” (data or text file) access request. Our protocols and

privacy protection goals/requirements are different for these two access request types. We will discuss our protocols in 6.2.1 and 6.2.2 for these two types.

6.2.1 Privacy Protection Protocol for Information Access Request

As we discussed before, Role's privacy protection is not desirable all the time. We can make it as an option for AR. So, AR will have an option to choose her access request as "Privacy Mode" or not. In most of cases, AR & SP can just follow traditional SOA access control standards to realize access control. When "Privacy Mode" option is selected (triggered by AR), SOA will work in a little bit different way (will discuss the details below) as illustrated in Figure 6.1.

Information Access request means that AR requests some data or text file which can be transferred to her. Figure 28 is the protocol we propose to protect AR's role information when it's regarded as privacy if the request is "information access request".

Figure 28: Privacy Protection Protocol for Information Access Request

Figure 28 illustrates our role privacy protection protocol for information access request. The protocol is also based IBE technology. SP encrypts required information with role's title as public key, and sends the cipher text to AR. Then AR will decrypt the cipher text with her credentials obtained from login and role assignment process in Chapter 5. If and only if AR holds private key of the Role, she can decrypt the cipher text and get the information. Now, let's check the detailed steps in this protocol:

Step 1: AR sends Access Request to SP. In this step, AR will also let SP know that if it's a role privacy preserving request or a normal request. This request will be implemented by improved XACML request language (will discuss in section 6.4);

Step 2: If it's a role privacy preserving request (this protocol discuss privacy preserving request only), SP will check from her policies to found out required roles for the information/message. Then, SP will create a random number S, and encrypt the random number S and messages by the required roles as public key:

$E_K(\text{Information}) \& E_K(S)$ (K is required role as public key such as "IT Manager".);

Step3: SP sends encrypted Message and random number S: $E_K(\text{Information}) \& E_K(S)$ as well as original random number S to AR;

Step4: AR decrypts $E_K(S)$ by her role credentials as private keys: $D_{R_i}^{-1}(E_K(S))$ (R_i = Role Credential), if there is one role credential K_i to let $D_{R_i}^{-1}(E_K(S)) = S$, that means the role credential of R_i is the required Role, then AR can decrypts $E_K(\text{Information})$ by the R_i :

$D_{R_i}^{-1}(E_K(\text{Information})) = \text{Information}$

Under the protocol, SP knows nothing about AR's role, however SP is sure that only when AR holds required Role(s), she can access the information. Both AR's role privacy and SP's information security are protected.

Some reader will point out that this is not a complete privacy preserving protocol. AR will get to know which role is required to access this information since she needs to decrypt and get the random number S first. If AR successfully decrypts and obtains the random number S, she knows she has permission for the required information. She can get to know not only the required information but also SP's required role as permission. However, as we discussed in section 6.1, this is not a huge privacy concern. For example, if AR's role is Junior Developer, by no matter what way she gets the information, she will get to know that her role as Junior Developer is enough for the information access. Or if she fails, it's very easy for her to assume that a more

senior role such as Senior Developer can have permission to access the information. Based on this nature, it's meaningless for us to be picky to SP's protect roles-to-permission policies. We already discussed it in details in section 6.1

Again, please note that, the protocol should be compatible and work well with SOA's current access control technologies and standards. We will discuss implementation later on.

6.2.2 Privacy Protection Protocol for Service Access Request

If what AR requires is access of a webpage, the situation will be difference from what we discussed in 6.2.1. It does not make sense for SP to send the webpage to AR, and AR needs access to view the webpage. In this case, AR is requesting access to some resource of SP rather than some detailed data or text file which could be transferred.

Our protocol in this situation is a different one.

Figure 29: Privacy Protection Protocol for Service Access Request

Figure 29 illustrates our role privacy protection protocol for service access request. It's also an IBE technology based protocol. SP created a random number and encrypts the number with role's title as public key, and sends the encrypted number to AR. Then AR will decrypt the cipher text with her credentials obtained from login and role assignment process in Chapter 5. Then AR will get a set of decrypted number $\{S_1, S_2, \dots, S_n\}$. If and only if AR holds private key of the required Role, she can get the number of S (even though AR does not know what's S and if there is a S in her value set). Now, let's check the detailed steps in this protocol:

Step 1: AR sends Access Request to SP. In this step, AR will also let SP know that if it's a role privacy preserving request or a normal request. This request will be implemented by improved XACML request language (will discuss in section 6.4);

Step 2: If it's a role privacy preserving request (this protocol discuss privacy preserving request only), SP will check from her policies to find out required roles for the information/message. Then, SP will create a random number S, and encrypt the random number S using the required roles as public key:

$E_K(S)$ (K is required role as public key such as "IT Manager".);

Step3: SP sends encrypted random number S: $E_K(S)$ to AR;

Step4: AR decrypts $E_K(S)$ by her role credentials as private keys: $S_i = D_{R_i}^{-1}(E_K(S))$ (R_i = Role Credential), and AR will get a set of decrypted value: $\{S_1, S_2, \dots, S_n\}$. She will sent the value set to SP for further process;

Step5: After receives values set from AR, SP will check if there is a value S in the set or not. If yes, she will grant access to AR or she will deny the access request.

In this protocol, AR has no knowledge about SP's policy, however, SP will get to know if AR's access permission is granted or declined. That is SP can know if AR has required role or not. This kind of knowledge disclosure is not able to be prevented. The service is hosted by SP, and she has to grant permission by herself and know if AR has permission to access or not. Even in this situation, SP just knows AR holds one of required roles to access the service. She does not know which exact role AR has and also she has no knowledge about other roles held by AR. This is already an accepted and satisfied result.

6.2.3 Role Hierarchy and Complicated Roles-to-Permission Policies

We already introduced RBAC (Role Based Access Control) in Chapter 1, and know that role hierarchy is one of its important features. Then how we handle role hierarchy in our protocols? We will still use the sample of Figure 4. We have three roles: IT Manager, Senior Developer, Junior Developer, among them, IT Manager is the most senior role and Junior Developer is most Junior Role. If there is information which is available for all three roles, then, in our protocols the information will only be encrypted by the most junior role – Junior Developer. The reason is the following:

Based on role assignment principle we discussed in section 5.2.5, if a senior role is assigned to AR, all junior roles belong to this senior role will automatically granted to AR. So, if AR holds a role of IT Manager she will for sure hold roles of Senior Developer and Junior Developer. So, if SP's permission can be assigned to all these three roles, SP just needs to encrypt information with the most junior role.

Role constraint is another important feature for RBAC. Since we already take constraint into consideration at Login and Role Assignment process, there is no user-to-role constraint and SPs will be responsible for their policies definition to prevent role-to-permission constraints.

Now, let's see how we handle some complicated roles-to-permission policies in our protocols. There are two kinds of complicated policies:

- Permission can be assigned to multiple roles;
- Permission can only be assigned to those who hold specific multiple roles

Then, how we handle these two kinds of complicated policies in our protocols? First, let me study the first kind "permission can be assigned to multiple roles".

For example, based on SP's policy, permission can be granted to two roles: "Senior Developer" & "Senior QA". Then, we just need to make a little bit change to our protocols.

For Information Access Request protocol (section 6.2.1), our modification is started from Step 2 in the following way:

Step 2: SP will create a random number S , and encrypt the random number S and messages by the two roles (Senior Developer and Senior QA) as public key:

$E_{K1}(\text{Information}) \& E_{K1}(S)$ ($K1$ is role "Senior Developer".);

$E_{K2}(\text{Information}) \& E_{K2}(S)$ ($K2$ is role "Senior QA".);

Then all these four encrypted values $E_{K1}(\text{Information})$ & $E_{K1}(S)$ and $E_{K2}(\text{Information})$ & $E_{K2}(S)$ will be sent to AR. The following steps are similar to original protocol. AR will decrypt $E_{K1}(S)$ and $E_{K2}(S)$ separately by her role credentials. AR just needs one role credential to decrypt S and get the information, so no matter she get S by which encrypted value, she can use the role credential decrypt $E_K(\text{Information})$ and get the information the process is accomplished.

For Service Access Request protocol (section 6.2.2), we still just need to modify the Step 2:

Step 2: SP will create a random number S , and encrypt the random number S by two roles (Senior Developer and Senior QA) as public key:

$E_{K1}(S)$ ($K1$ is role "Senior Developer".);

$E_{K2}(S)$ ($K2$ is role "Senior QA".);

After AR receives the two encrypted values, she will decrypt them separately by all her role credentials, and then she gets two values set:

$\{S_1, S_2, \dots S_n\}_{K1}$ and $\{S_1, S_2, \dots S_n\}_{K2}$

SP will check with the two value set, if she find S from no matter which value set, the permission will be granted to AR.

Now, let's look into situation of "permission can only be assigned to those who holds multiple roles". For example, permission can only be assigned to a person who holds both the following roles: Senior Developer & Developer in Project One Team.

For Information Access Request protocol (section 6.2.1), we modify protocol from step 2 (improved Information Access Request protocol under: "permission can only be assigned to those who holds multiple roles" scenario)

Step 2: SP will create a random number $S1$, and encrypt the random number $S1$ and messages by one of required role as public key first:

$E_{K1}(\text{Information})$ & $E_{K1}(S1)$ ($K1$ is one of required role of "Senior Developer".);

Then SP create another random number $S2$, and encrypt $S2$ and $E_{K1}(\text{Information})$ by the other required role as public key, and she will get:

$E_{K2}(E_{K1}(\text{Information}))$ & $E_{K2}(S2)$ ($K2$ is the other required role of "Developer in Project One Team".

Step3: SP sends $E_{K2}(E_{K1}(\text{Information}))$, $E_{K1}(S1)$, $E_{K2}(S2)$, $S1$ and $S2$ to AR;

Step4: AR decrypts $E_{K1}(S1)$ by her role credentials as private keys: $D_{Ri}^{-1}(E_{K1}(S1))$ (Ri = Role Credential), if there is one role credential $R1$ to let $D_{R1}^{-1}(E_{K1}(S1)) = S1$, that means the role credential of $R1$ is one of the required Role, then AR will keep this role credential and try to decrypt $E_{K2}(S2)$ by the same process. If she can successfully decrypt $E_{K2}(S2)$ by $R2$, then AR can decrypts $E_{K2}(E_{K1}(\text{Information}))$ by the $R1$ and $R2$:

$$D_{R1}^{-1}(D_{R2}^{-1}(E_{K2}(E_{K1}(\text{Information})))) = \text{Information}$$

For Service Access Request protocol (section 6.2.2), we also modify protocol from step 2 (improved Service Access Request protocol under: “permission can only be assigned to those who holds multiple roles” scenario):

Step 2: SP will create a random number $S1$, and encrypt the random number $S1$ by one of the required roles (Senior Developer) as public key:

$$E_{K1}(S1) \text{ (K1 is one of the required role “Senior Developer”).}$$

Then SP create another random number $S2$, and encrypt $S2$ by the other required role “Developer in Project One Team” as public key, and she will get:

$$E_{K2}(S2) \text{ (K2 is the other required role of “Developer in Project One Team”).}$$

Step3: SP sends encrypted random numbers $E_{K1}(S1)$ $E_{K2}(S2)$ to AR;

Step4: AR decrypts $E_{K1}(S1)$ and $E_{K2}(S2)$ separately by her role credentials as private keys:

$$S1_i = D_{Ri}^{-1}(E_{K1}(S1)) \text{ (R}_i \text{ = Role Credential);}$$

$$S2_i = D_{Ri}^{-1}(E_{K1}(S2)) \text{ (R}_i \text{ = Role Credential);}$$

AR will get two set of decrypted value: $\{D_1, D_2, \dots D_n\}$. and $\{E_1, E_2, \dots E_n\}$. She will send the two value set to SP for further process;

Step5: After receives values set from AR, SP will check if there is a value $S1$ in the set $\{D_1, D_2, \dots D_n\}$ and $S2$ in the set $\{E_1, E_2, \dots E_n\}$. If yes, she will grant access to AR or she will deny the access request.

The purpose of this section is to present the flexibility of our protocols. Policies and access control scenario in industry could be much more complicated, however, our protocols can be flexible to be modified to meet requirements.

6.3 Implementation of our protocols in Access Request & Response

We already provided a detailed description for SOA access control standards in Chapter 4. As we know, XACML is used as SOA access control standard and it includes Policy, Request and Response languages. In addition, SOA has lots of other standards to support its technologies, for example, SOAP is used as information transfer standards in SOA. As we mentioned several times in this thesis, one of major challenges we face is how to implement our protocols using current SOA standards. We may make reasonable modifications to these standards, and add new functions (such as cryptography functions). However, our implementations have to be based on current standards which are popularly accepted by SOA industry solutions. In this section, we will discuss on implementations of our protocols in Access Request & Response process.

6.3.1 Implementation of Information Access Request Protocol

As we described in Section 6.2.1, one of the most important innovation of our protocol is that AR has the option to set her request as a normal request mode (traditional SOA access request without privacy protection), and privacy protection mode in Step 1. The normal mode will go with the normal method that we discussed in Section 4.1 and Section 4.2. If AR chooses to use role privacy protection mode, it will be implemented with a different way which is still compatible with SOA standards. We will provide a simple sample to explain the implementation of these two options:

```
<Request>
  <Subject>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
      DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name">
      <AttributeValue>sample@users.example.com</AttributeValue> </Attribute>
    <Attribute AttributeId="role" DataType="http://www.w3.org/2001/XMLSchema#string"
      Issuer="admin@users.example.com">
      <AttributeValue>Senior Developer</AttributeValue></Attribute>
    </Subject>
    <Resource>
      <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
        DataType="http://www.w3.org/2001/XMLSchema#anyURI">
      <AttributeValue>developer-guide.doc</AttributeValue></Attribute>
    </Resource>
    <Action>
      <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
        DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>read</AttributeValue></Attribute>
    </Action>
  </Request>
```

Figure 30: XACML Request In Tradition Way

```

<Request>
  <Subject>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
      DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name">
      <AttributeValue>sample@users.example.com</AttributeValue> </Attribute>
    <Attribute AttributeId="role" DataType="http://www.w3.org/2001/XMLSchema#string"
      Issuer="admin@users.example.com">
      <AttributeValue>Privacy Mode</AttributeValue></Attribute>
    </Subject>
  <Resource>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
      DataType="http://www.w3.org/2001/XMLSchema#anyURI">
      <AttributeValue>developer-guide.doc</AttributeValue></Attribute>
    </Resource>
  <Action>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
      DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>read</AttributeValue></Attribute>
    </Action>
  </Request>

```

Figure 31: XACML Request under Privacy Mode

There are four key elements in this sample:

- This access request is under RBAC Profile of XACML, so it's Role-Based Access Control;
- AR holds a role of Senior Developer;
- Requested access is for a document of "developer-guide.doc" in the individual system;
- Requested action for the resource is "read".

Figure 30 and Figure 31 are the implementation of traditional SOA access request and privacy protection mode accordingly. In Figure 30, we can see that role title of AR is listed in the request code, however, in Figure 31, there is no role title displayed. In stead, there is a key word of "Privacy Mode" in the place of role title. When SP sees the key word (it's a standard agreed by all individual systems within SOA), she will enter Privacy Mode process.

So, after SP receives the request, she can identify the request is in "Privacy Mode", then her response will be different. She will check her policy set to find out which roles are eligible to read the information of the resource. In Chapter 4, we provided policy samples in Figure 12, Figure 13, Figure 14 and Figure 15.

We can see that these sample policies are all Role PolicySets which are suitable for the situation that SP knows AR's role ready and try to search permission policies by the role. However, in our privacy protection mode, role information is hidden. SP only knows the requested "resource" title. So, we need to create a set of Resource PolicySets which is in the similar format of Role PolicySets:

```
<PolicySet PolicySetId="Resource PolicySet for developer-guide.doc"
  CombiningAlgorithm="permit-overrides">
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="string-match">
          <AttributeValue> developer-guide.doc</AttributeValue>
          <SubjectAttributeDesignator AttributeId="Role"/>
        </ResourceMatch>
      </Resource>
    </Resources>
    <Subjects>
      <AnySubject/>
    </Subjects>
    <Actions>
      <AnyAction/>
    </Actions>
  </Target>
  <PolicySetIdReference>Permission PolicySet for developer-
guide.doc </PolicySetIdReference>
</PolicySet>
```

Figure 32: Sample – Resource PolicySet for developer-guide.doc

```

<PolicySet PolicySetId="Permission PolicySet for developer-guide.doc"
  CombiningAlgorithm="permit-overrides">
  <Target>
    <Policy PolicyId="Permissions specifically for developer-
guide.doc" CombiningAlgorithm="permit-overrides">
      <Target>
        <Subjects><AnySubject/></Subject>
        <Resources><AnyResource/></Resources>
        <Actions><AnyAction/></Actions>
      </Target>
      <Rule RuleId="Permission to Senior Developer" Effect="Permit">
        <Target>
          <Subjects>
            <AnySubject/>
          </Subjects>
          <Actions>
            <Action>
              <ActionMatch MatchId="string-match">
                <AttributeValue>Read</AttributeValue>
                <ActionAttributeDesignator AttributeId="action-id"/>
              </ActionMatch>
            </Action>
          </Actions>
        </Target>
      </Rule>
    </Policy>
  </PolicySet>

```

Figure 33: Sample – Permission PolicySet for developer-guide.doc

From Figure 32 (Resource PolicySet for developer-guide.doc) and Figure 33 (Permission PolicySet for developer-guide.doc), SP gets to know that role of “Senior Developer” will be granted read permission.

Then, SP will encrypt the information (document of developer-guide.doc) by the “Senior Developer” as public key. She also creates a random number N, and encrypts N by the “Senior Developer” as public key. Please be noted that these encryption operations are handled at SP as an individual system, so it’s not a SOA related. We can regard them as single machine operation and it can be easily implemented and it’s not the topic of our discussion. Then, SP sends encrypted information, encrypted value N and original N to AR. All values are transferred via SOAP which is file/data transfer standard for SOA and web service. Figure 34 is a sample of such SOAP code to transfer encrypted information, encrypted random number N and original random number N.

```

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/";
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance";
  xmlns:xsd="http://www.w3.org/2001/XMLSchema";>
  <soap:Body>
    <GetInformationResponse xmlns="http://www.xmlforasp.net">
      <GetInformationResult>
        <!-- Encrypted document of developer-guide.doc -->
        <EncryptedInformation>
          grIKIJMCSYHrgXIRThnxEYqZicqWeio0OJ3
          p+8NzFuqxxA8Y155qaN/iy1Ywmm86fwqFmP
          8HL4/8IRA9dlfMySAkB5MF1KyEv5ReConcE
          DLoyl4sXJiYgWPQceh4XF06r49PkQGk8mnb
          WlpRbiiTJ76Uk22gCjdiU5IcWHnzB3k=
        </EncryptedInformation>
        <!-- Encrypted Challenge Number N -->
        <EncryptedNumber>
          wDz/BvGUIJwL6WXNsc2/FGXiG9tlW4818VP
          wzlOSetiCSSz7kw4jwp1QvDJhJ+tr78X1uT
          zPkOQUbrUjHjaVnEwyP/Ez/uqVX7WW5zmvA
          y3ZtPmkkzHlJnM8f+FyRMG6Fr6nzZ/ZDEw6
          s+Vai5LTTLs3JZ297i5XTMAsaITgc74=
        </EncryptedNumber>
        <!-- Original Challenge Number -->
        <ChallengeNumber>
          1234567890
        </ChallengeNumber>
      </GetInformationResult>
    </GetInformationResponse>
  </soap:Body>
</soap:Envelope>

```

Figure 34: Sample of Encrypted SOAP Message in Information Access Request

After AR receives the encrypted message, she will decrypt “EncryptedNumber” by one of her role credentials first. If the decrypted value matches “ChallengeNumber”, then use the credential to decrypt “EncryptedInformation” to get the information she needs. If it fails, she will try another role credential until the last. If AR does not own the required role credential, she will know nothing about the information. Again, the decryption and all other related operations are handled at AR as an individual member of SOA.

Our protocol put some new functions/algorithms to traditional SOA such as: options for AR to setup “Privacy mode” or stay with normal access request; and a bunch of Resource PolicySet and Permission PolicySet for Resource are created.

These functions are easy to be implemented by logic or programming point of view. Most important, the solution is still within SOA and web service basic standards and protocols such as XACML and SOAP. So, there is no problem for implementation of our protocol.

For all encryption and decryption operations, they are handled at individual systems, and of course there is no compatible or implementation issue for these operations.

6.3.2 Implementation of Service Access Request Protocol

Implementation of Service Access Request Protocol is similar to Information Access Request Protocol. Especially access request, Resource PolicySet, and Permission PolicySet, these codes are the same as code in implementation of Information Access Request, and the only difference is that the resource in these code is a service such as a webpage rather than a transferable information.

```
<PolicySet PolicySetId="Resource PolicySet for  
http://server.example.com/code/docs/developer-guide.html "  
  CombiningAlgorithm="permit-overrides">  
  <Target>  
    <Resources>  
      <Resource>  
        <ResourceMatch MatchId="string-match">  
          <AttributeValue>  
            http://server.example.com/code/docs/developer-guide.html </AttributeValue>  
          <SubjectAttributeDesignator AttributeId="Role"/>  
        </ResourceMatch>  
      </Resource>  
    </Resources>  
    <Subjects>  
      <AnySubject/>  
    </Subjects>  
    <Actions>  
      <AnyAction/>  
    </Actions>  
  </Target>  
  <PolicySetIdReference>Permission PolicySet for  
http://server.example.com/code/docs/developer-guide.html  
</PolicySetIdReference>  
</PolicySet>
```

Figure 35: Sample – Resource PolicySet for Service

In this section, we only provide a sample of Resource PolicySet for Service. Please check Figure 35, and you can find that the “developer-guide.doc” as the resource title is changed to a non-transferable web page link: <http://server.example.com/code/docs/developer-guide.html>

In Step 1 of the protocol, AR choose “Privacy Mode” and send SP the request, the request code is the same as what we used in implementation of Information Access Request. (resource is changed to a service in this case)

In Step 2, SP will check Resource PolicySet and Permission PolicySet (as the samples in last section) and create a random S. If SP finds that the matched role is “Senior Developer”, she will encrypt S by “Senior Developer” as public key.

In Step 3, SP will send the encrypted S to AR via SOAP (as described in Figure 36 as a sample):

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/";
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance";
  xmlns:xsd="http://www.w3.org/2001/XMLSchema";>
  <soap:Body>
    <GetInformationResponse xmlns="http://www.xmlforasp.net";>
      <GetInformationResult>
        <!-- Encrypted Challenge Number S -->
        <EncryptedNumber>
          wDz/BvGULjwL6WXNsc2/FGXiG9tlW4818VP
          wzlOSetiCSSz7kw4jwp1QvDJhJ+tr78X1uT
          zPkOQUbrUjHjaVnEwyP/Ez/uqVX7WW5zmvA
          y3ZtPmkkzHIJnM8f+FyRMG6Fr6nzZ/ZDEw6
          s+Vai5LTTLs3JZ297i5XTMAsaITgc74=
        </EncryptedNumber>
      </GetInformationResult>
    </GetInformationResponse>
  </soap:Body>
</soap:Envelope>
```

Figure 36: Sample of Encrypted SOAP Message to Transfer Encrypted S

In Step 4, AR will decrypt the encrypted S by all role credentials she holds. Please be noted that this process is handled at AR as an individual system or client, and it's not SOA related.

In Step 5, AR will send the value set $\{S_1, S_2, \dots, S_n\}$ she gets at Step 4 back to SP via SOAP (Figure 37):

```

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/";
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance";
  xmlns:xsd="http://www.w3.org/2001/XMLSchema";>
  <soap:Body>
    <GetInformationResponse xmlns="http://www.xmlforasp.net";>
      <GetInformationResult>
        <!-- Decrypted Challenge Number S1 -->
        <DecryptedNumber1>
          32342
        </DecryptedNumber1>
        <!-- Decrypted Challenge Number S2 -->
        <DecryptedNumber2>
          42323
        </DecryptedNumber2>
        ...
        <!-- Decrypted Challenge Number Sn -->
        <DecryptedNumbern>
          23212
        </DecryptedNumbern>
      </GetInformationResult>
    </GetInformationResponse>
  </soap:Body>
</soap:Envelope>

```

Figure 37: Sample of Encrypted SOAP Message to Transfer Encrypted {S1, S2, ... Sn}

In Step 6, AR will review received {S₁, S₂, ... S_n} and compare her original random number of S to decide to permit or deny and sends response back to AR (Figure 38):

```

<Response>
  <Result>
    <Decision>Permit</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
    </Status>
  </Result>
</Response>

```

Figure 38: Sample of Response in Service Access Request

The response code is exactly the same as traditional XACML response language code.

We can conclude that implementation of Service Access Request can also take use of current XACML standards just like implementation of Information Access Request. Also, the encryption and decryption operations in Step 2 and Step 4 are handled as individual systems which do not belong to SOA topics and we will not discuss implementation of these two steps in this thesis.

6.4 Security Review on Protocols for Access Request & Response Process

In this Section, we will review what kind of privacy protection we can provide in our protocols for Access Request & Response Process. As we discussed in Section 6.1, there is a tradeoff between privacy protection and system performance, and the major factor we decide on the tradeoff is importance and sensitive level of privacy. From analysis we made in Section 6.1, AR's roles privacy and SP's permission-to-role policies are not as sensitive as AR's attributes privacy and RA's attributes-to-role policies. The purpose of our research work is to provide the best and satisfying solutions to protect privacy which in the same time are compatible with current SOA standards and technologies.

First, let's see how AR's role privacy is protected:

AR can choose to set her role as privacy (need to be protected) or not (do not need to protect role privacy). Our protocols will only be used when AR chooses "Privacy Mode", or SOA will go with traditional access process.

In Scenario of Information Access Request, SP will encrypt required information by required role as public key and send it to AR, in this case: first SP has ZERO knowledge of AR's role. After SP sends out encrypted information, she will not be involved in decrypted process. So, SP will not know if AR can get the information or not, and it is not possible for SP get any information about AR's role information; second SP is sure that her information will not be viewed by anyone without a valid role credential. This is information security issue, and it's the basic requirement of our protocol. Based on IBE [2][4][6][37][38], if the information is encrypted by role's title string as public key, only those who holds valid role credential as private key can decrypt it. The role credential can only be obtained through Login & Role assignment process described in Chapter 5.

In Scenario of Service Access Request, SP has to get involved to decide if permit should be granted to AR or not, and as a result, SP will get to know at least that if AR's request is permitted or not. If the request is permitted, SP will get to know that AR holds at least one required roles.

This knowledge disclosure is not possible to be avoided. If we check our protocol, we know that SP will check S values from decrypted value set of $\{S1, S2, \dots, Sn\}$, and if there is a S in the set, then SP knows AR holds required Role, and grants permission. However, it's still acceptable for privacy protection because of the following reasons.

First, SP knows AR holds a required role, but she does not know anything about all other roles held by AR, and it is already huge improvement comparing with traditional RB_XACML technology.

Second, based on our discussion in Section 6.1, AR's role information is not a privacy sensitive, and it will not cause serious subsequence for some role information disclosure.

Third, combining with privacy protocols in Login & Role Assignment process proposed in this thesis, the whole privacy protection is satisfying and acceptable.

Now, let's review SP's policy privacy protection in this process.

In Scenario of Information Access Request, SP encrypts required information by required role and sends encrypted value to AR, then AR will decrypt the value by her role credentials. If AR is able to decrypt the random number and information by some role, she will know this role is required by the policy, however, it is still an acceptable result, because:

First, AR will get to know which role credential let her get the permission of access. However, this does not mean that she knows the permission policies. As we discussed in Section 6.2.3, SP's policies can let several roles have the permission, and she will just encrypt the information by the most junior role. So, AR can not be sure about what are SP's policies;

Second, in Section 6.1, we analyzed sensitive and importance level of SP's permission-to-role policies, and could know that thesis policies are not privacy sensitive

In Scenario of Service Access Request, AR receives decrypted random number S, and decrypts it by all her role credentials and get a value set of $\{S1, S2, \dots, Sn\}$, this value set will be sent to SP, SP will check and value and make decision to grant permit or deny. So, AR does not get involved in any policies related operations. As a result AR knows nothing about SP's policies.

In conclusion, we provide privacy protection option for AR, and protect both AR's role privacy and SP's policies in a reasonable and acceptable way. Considering the implementation, system efficiency, compatibility etc. factors, privacy protection in this process is not designed to "ZERO" knowledge disclosure. However, it is already a great improvement comparing with traditional

RB_XACML standards and it is a good supplementary to the whole privacy solution we proposed in this thesis as a part of the whole solution.

7. Conclusion and Future Work

In this thesis, we worked on providing privacy protection solution for RBAC in SOA environment. We proposed a new SOA RBAC solution which meets single sign on requirement and is privacy preserving and easier to be maintained. This RBAC solution is composed of two processes: Login & Role Assignment process and Access Request & Response process. Login & Role Assignment is privacy sensitive, and our protocol for this process is ZERO knowledge disclosure which means both AR and RA knows nothing about the counterpart's privacy. Login & Role Assignment process can be divided into two stages: attributes values exchanging and policy mapping. We proposed three protocols/solutions for each stage with different privacy protection and complication level, and at least one of these protocols/solutions meets the goal of ZERO knowledge disclosure or complete privacy protection. Access Request & Response process is regarded as less privacy sensitive and our solution must be compatible with current SOA standards. In SOA, there are two kinds of access requests: Information Access and Service Access. We provided AR option to setup privacy mode and gave two different protocols to handle privacy protection for these two kinds of access requests. It's not ZERO knowledge disclosure protocols but based on our security and privacy analysis, our protocols are good enough to provide necessary privacy protection. It already greatly improved privacy protection comparing with traditional SOA access control standards and technologies.

As a whole, we made a thorough analysis on SOA RBAC solution and provided a set of new protocols/solutions which make the whole process privacy friendly and are also beneficial for security policy management and system maintenance.

Next, we will continue to work on how to improve current SOA standards such as XACML to make it possible to realize ZERO knowledge disclosure during the Access Request & Response process without obvious impact system's performance and efficiency.

References

- [1] T. Yu, M. Winslett, K. E. Seamons, "Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation", *ACM Trans. On Information and System Security*, 6(1):1-42, February 2003
- [2] N. Li, W. Du, D. Boneh, "Oblivious Signature-Based Envelope", *Distributed Computing*, vol. 17, no. 4, pp. 293-302, 2005
- [3] J. Li, N. Li, "OACerts: Oblivious Attribute Certificates", *IEEE Trans. On Dependable and Secure Computing*, vol. 3, no. 4, October-December 2006
- [4] K. Frikken, M. Atallah, J. Li, "Attribute-Based Access Control with Hidden Policies and Hidden Credentials", *IEEE Trans. On Computer*, vol. 55, No. 10, October 2006
- [5] D. Yao, R. Tamassia, "Compact and Anonymous Role-Based Authorization Chain", *ACM Trans. On Information and System Security*, vol. 12, no. 3, article 15, January 2009
- [6] D. Boneh, M. Franklin, "Identity-Based Encryption from the Weil Pairing", *SIAM J. of Computing*, vol. 32, No. 3, pp. 586-615, 2003
- [7] D. Boneh, C. Gentry, B. Lynn, H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps", *Proceedings of the international conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt 03)*. 416-432, 2003
- [8] M. K. Raghavendra, "An Architectural Approach to RBAC Policy Management Framework in a SOA driven Enterprise", unpublished
- [9] D. W. Chadwick, A. Otenko, E. Ball, "Role-Based Access Control With X.509 Attribute Certificates", *IEEE Computer Society*, 1089-7801, 2003
- [10] Z. Qu, N. Meng, "Design and Implementation of the RBAC-SOA Model", ICSP 2008 Proceedings
- [11] F. Xu, G. Lin, H. Huang, L. Xie, "Role-based Access Control System for Web Service", *Proceedings of the 4th International Conference on Computer and Information Technology (CIT 04)*, 2004
- [12] "Core and hierarchical role based access control (RBAC) profile of XACML v2.0, OASIS
- [13] "eXtensible Access Control Markup Language (XACML) Version 2.0, OASIS Standard, February 1, 2005
- [14] "SAML 2.0 profile of XACML v2.0", OASIS Standard, February 1, 2005
- [15] "Reference Model for Service Oriented Architecture 1.0", Committee specification 1, 2, August 2006
- [16] "Web Service Architecture", W3C, <http://www.w3.org/TR/ws-arch/>
- [17] Wikipedia, the free encyclopedia, <http://en.wikipedia.org/wiki/>

- [18] B. Thuraisingham, "Security for Web Services and Service Oriented Architectures", Lecture
- [19] "An Introduction To Role-Based Access Control", NIST/ITL Bulletin, December, 1995
http://csrc.nist.gov/groups/SNS/rbac/documents/design_implementation/Intro_role_based_access.htm
- [20] A. Oluwaseyi, "Service Oriented Architecture & Web Services", Master Degree Thesis in Computer Science, January, 2008
- [21] F. Meier, "Service Oriented Architecture Maturity Models: a guide to SOA Adoption", Master Degree Thesis in Computer Science, 2006
- [22] "Service Oriented Architecture Security Vulnerabilities – Web Services", Systems and Network Analysis Center www.nsa.gov/snac
- [23] V. Jonnaganti, "An Integrated Security Model for the Management of SOA", Master Degree Thesis in Software Engineering and Management
- [24] "SOA Security (red book)", IBM
- [25] J. Fiere, "SOA Security", Master Degree Thesis in Information Sciences, November 2007
- [26] N. A. Nordbotten, "XML and Web Services Security Standards", *IEEE Communications Surveys & Tutorials*, vol. 11. No. 3, third quarter 2009
- [27] R. Bhatti, J. B. D. Joshi, E. Bertino, A. Ghafor, "Access Control in Dynamic XML-based Web-Services with XRBAC", in processing of The First International Conference on Web Service, Las Vegas, June 23-26, 2003
- [28] UDDI Version 3.0 Published Specification, July 19, 2002, http://uddi.org/pubs/uddi_v3.htm
- [29] W3C Note. Web Services Description Language (WSDL) 1.1, March 15, 2001,
<http://www.w3.org/TR/wsdl>
- [30] Y. Demchenko, "Using XACML and SAML for Authorisation messaging and assertions: XACML and SAML standards overview and usage examples", March 28, 2005
- [31] E. Dubuis, B. Fachhochschule, "XACML: An Introduction", January 2008, Lecture
- [32] K. Kim, H. J. Ko, W. G. Choi, E. J. Lee, U. M. Kim, "A Collaborative Access Control based on XACML in Pervasive Environments", *IEEE International Conference on Convergence and Hybrid Information Technology*, 2008
- [33] A. Menezes, P. van Oorschot, S. Vanstone, "Handbook of Applied Cryptography", *CRC Press*, 1996
- [34] B. Barak, "Oblivious Transfer (OT) and Private Information Retrieval (PIR)", November 29, 2007
- [35] R. Shahrinian, T. Hu, "Foundations of Cryptography", March 16, 2005

- [36] W. Tzeng, "Efficient 1-Out-of-n Oblivious Transfer Schemes with Universally Usable Parameters", *IEEE TRANSACTIONS. ON Computers*, vol. 53, no. 2. February 2004
- [37] A. Shamir, "Identity-Based Cryptosystems and Signature Schemes", *Advances in Cryptology – Proc. CRYPTO 1984*, pp. 47-53, 1984
- [38] C. Cocks, "An Identity Based Encryption Scheme Based on Quadratic Residues", *Proc. Eighth IMA International Conference Cryptography and Coding*, pp. 360-363. December, 2001
- [39] X. Boyen, "An Introduction to Identity-Based Encryption",
<http://ai.stanford.edu/~xb/qut07a/slides/index.html>
- [40] I. Damgard and M. Jurik, "A Generalisation, a Simplification and Some Applications of Pailier's Probabilistic Public-Key System", *Proc. Fourth International workshop Practice and Theory in Public Key Cryptosystems*, pp. 119-136, 2001
- [41] T. Okamoto, S. Uchiyama, E. Fujisaki, *Epoc: Efficient Probabilistic Public-key Encryption Submission to IEEE p1363a.*, 1998
- [42] P. Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes", *Advances in Cryptology – Proc. EUROCRYPT 1999*, pp. 223-238, 1999
- [43] M. Freeman, K. Nissim, B. Pinkas, "Efficient Private Matching and Set Intersection", *Advances in Cryptology – Proc. EUROCRYPT 2004*, pp. 1-19, May 2004
- [44] T. Namli, A. Dogac, "Using SAML and XACML for Web Service Security & Privacy", Not Published, Software R&D Center, Middle East Technical University
- [45] D. Wahlin, "Encrypt SOAP Message Parameters using Custom Attributes and Headers",
<http://www.dotnet247.com/247reference/messages/24/121010.aspx>
- [46] C. A. Ardagna, S. Vimercati, S. Paraboschi, E. Pedrini, P. Samarati, "An XACML-Based Privacy-Centered Access Control System", *ACM 978-1-60558-787-5/09/11*, 2009
- [47] J. Crampton, "XACML and Role-Based Access Control", *DIMACS Workshop on Secure Web Services and e-Commerce*
- [48] "Sun's XACML Implementation Programmer's Guide for Version 1.2",
<http://sunxacml.sourceforge.net/guide.html>
- [49] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, C. E. Youman, "Role-Based Access Control Models", *IEEE, 0018-9162/96*, 1996
- [50] G. Geethakumari, A. Negi, V. N. Sastry, "A Cross – Domain Role Mapping and Authorization Framework For RBAC in Grid System", *International Journal of Computer Science and Application © 2009 Technomathematics Research Foundation Vol. 6 No. 1, pp. 1-12*, 2009

- [51] S. Weber, P. Austel, M. McIntosh, "A Framework for Multi-Platform SOA Security Analyses", *IEEE International Conference on Web Services (ICWS 2007)* 0-7695-2924-0/07, 2007
- [52] H. Amouzegar, S. Mohammadi, M. J. Tarokh, A. N. Hidaji, "A New Approach on Interactive SOA Security Model Based on Automata", *IEEE* 978-0-7695-3131-1/08, 2008
- [53] A. H. Karp, "Authorization-based Access Control for the Service Oriented Architecture", *IEEE Published in the Fourth International Conference on Creating, Connecting, and Collaborating through Computing (C5)*, 26-27, January 2006,
- [54] P. Pradat, M. Vallot, "Securing SOA and Traditional Infrastructure", *A Bull Evidian White Paper*, 2005
- [55] F. Meier, "Service Oriented Architecture Maturity Models: A guide to SOA Adoption", Masters Dissertation in Computer Science, 2006
- [56] Z. Y. Peng, S. P. Li, "Privacy Protection in Trust Management", *IEEE* 978-1-4244-2096-4/08, 2008
- [57] J. Brickell, V. Shmatikow, "Privacy-preserving Graph Algorithms in the Semi-honest Model", *International Association for Cryptologic Research B. Roy (Ed.): ASIACRYPT 2005, LNCS 3788*, pp. 236-252, 2005
- [58] A. Damodaram, H. Jayasri, "Authentication without Identification Using Anonymous Credential System", *International Journal of Computer Science and Information Security IJCSIS Vol. 3 No.1*, July 2009
- [59] R. Bhaskar, K. Chandrasekaran, S.V. Lokam, P. L. Montgomery, R. Venkatesan, Y. Yacobi, "Vulnerabilities in Anonymous Credential Systems", *STM*, 2007
- [60] P. Mazzoleni, E. Bertino, B. Crispo, S. Sivasubramanian, "XACML Policy Integration Algorithms", *ACM* 1-59593-354-9/06/0006, 2006
- [61] J. S. Park, R. Sandhu, "Role-based Access Control on the Web", *ACM Transactions on Information and System Security*, Vol. 4, No. 1, 2001
- [62] G. Swart, B. Aziz, S. N. Foley, J. Herbert, "Trading off Security in a Service Oriented Architecture", *IBM Almaden Research*, Not Published

Glossary

Term	Description
AR	Access Requestor
Cryptography	<p>Cryptography (or cryptology; from Greek κρυπτός, <i>kryptos</i>, "hidden, secret"; and γράφω, <i>gráphō</i>, "I write", or -λογία, <i>-logia</i>, respectively) is the practice and study of hiding information. Modern cryptography intersects the disciplines of mathematics, computer science, and engineering. Applications of cryptography include ATM cards, computer passwords, and electronic commerce</p> <p>In cryptography, encryption is the process of transforming information (referred to as plaintext) using an algorithm (called cipher) to make it unreadable to anyone except those possessing special knowledge, usually referred to as a key. The result of the process is encrypted information (in cryptography, referred to as ciphertext). In many contexts, the word encryption also implicitly refers to the reverse process, decryption (e.g. "software for encryption" can typically also perform decryption), to make the encrypted information readable again (i.e. to make it unencrypted).</p>
Encryption & Decryption	
IBE	An Identity Base Encryption (IBE) scheme is a public-key cryptosystem where any string is a valid public key. In particular, email addresses and dates can be public keys.
Oblivious Transfer	In cryptography, an oblivious transfer protocol (often abbreviated OT) is a protocol by which a sender sends some information to the receiver, but remains oblivious as to what is received.
Public Key & Private Key	Public Key & Private Key cryptography is a cryptographic approach which involves the use of asymmetric key algorithms instead of or in addition to symmetric key algorithms. Unlike symmetric key algorithms, it does not require a secure initial exchange of one or more secret keys to both sender and receiver. The asymmetric key algorithms are used to create a mathematically related key pair: a secret private key and a published public key. Use of these keys allows protection of the authenticity of a message by creating a digital signature of a message using the private key, which can be verified using the public key. It also allows protection of the confidentiality and integrity of a message, by public key encryption, encrypting the message using the public key, which can only be decrypted using the private key.
PV	Policy Verifier
RA	Role Assigner
RBAC	role-based access control (RBAC) is an approach to restrict system access to authorized users in which access permission is assigned based role of the access requestor.

SAML	Security Assertion Markup Language (SAML) is an XML-based standard for exchanging authentication and authorization data between security domains, that is, between an <i>identity provider</i> (a producer of assertions) and a <i>service provider</i> (a consumer of assertions). SAML is a product of the OASIS Security Services Technical Committee.
Single Sign On	Single sign-on (SSO) is a property of access control of multiple, related, but independent software systems. With this property a user logs in once and gains access to all systems without being prompted to log in again at each of them
SOA	service-oriented architecture (SOA) is a flexible set of design principles used during the phases of systems development and integration. A deployed SOA-based architecture will provide a loosely-integrated suite of services that can be used within multiple business domains.
SOAP	SOAP, originally defined as Simple Object Access Protocol, is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks. It relies on Extensible Markup Language (XML) as its message format, and usually relies on other Application Layer protocols (most notably Remote Procedure Call (RPC) and HTTP) for message negotiation and transmission
SP	Service Provider
XACML	XACML stands for Extensible Access Control Markup Language. It is a declarative access control policy language implemented in XML and a processing model, describing how to interpret the policies.
XML	XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form.