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

Helen Cheung, Celia Li, Ye Yu, Cungang Yang

Department of Electrical and Computer Engineering
Ryerson University
Toronto, ON, Canada

ABSTRACT

Service Oriented Architecture (SOA) changes the way of conducting business by opening their services to the larger business world over the networks. However, the "open" and "interoperable" properties of SOA make privacy a sensitive security issue. In SOA, service providers (SPs) limit permission of access to specific authorized Access Requestors (ARs). SPs need to verify ARs' identity information, but ARs may not willing to disclose their privacy to unknown SPs in an open system. To solve this conflict in SOA environment, we propose privacy preserving protocols for role-based access control (RBAC) in the SOA environment. The security analysis demonstrates that our protocols are privacy protected. Moreover, the implementation of the proposed protocols are compatible with current SOA standards and technologies such as XACML and SOAP.

KEYWORDS

Privacy Protection, Service Oriented Architecture, Role-based Access Control, XACML.

1. INTRODUCTION

Service Oriented Architecture (SOA)[1] is a paradigm for organizing and utilizing distributed capabilities that are under the control of different ownership domains. SOA consists of multiple autonomous systems that communicate through networks. SOA is loose coupled: sub-systems under SOA do not communicate with each other directly. Instead, they communicate through services. Core components of SOA include service consumer, service provider and service broker

Recently, security is becoming a major concern for SOA popularization and promotion[2][3][4]. Organizations and IT industry giants such as W3C, WS-I, OASIS, IBM are working on SOA security architecture, standards and protocols. The SOA security framework[5] provides us a clear picture of major achievements in SOA security research and implementations. However, this framework does not provide identity privacy protection. Since SOA is loosely coupled by individual systems, clients are not willing to disclose their identity information (business identity or personal identity) to unknown SPs. The lack of identity privacy protection of current SOA security framework is the main motivation of this research works.

In SOA, RB-XACML is a modified version of traditional XACML in which XACML policy meet features of RBAC, such as roles, permission to role assignments etc. RB-XACML has the following weaknesses:

DOI : 10.5121/ijnsa.2014.6301

- No authoritative and recognized organization provide unified roles, attributes definitions and attributes-to-roles assignment rules. As a result, it causes lots of coordination and maintenance work.
- The user-role assignment is based on attributes and a RA's (Role Assigner) role is an "attribute value". In SOA, ARs' attributes are transported between sub-systems or components, which is not a privacy friendly method because an AR may not want its role to be disclosed to unknown SPs either.

To solve the above problems, RBAC [6][7] in SOA is divided into two separate processes: login & role assignment, and access request & response. Login & role assignment process is handled by a independent system apart from traditional SOA structure. Access request & response process is still operated within SOA. This approach simplifies the system maintenance work.

In addition, we provide privacy preserving protocols for both processes. The privacy protection solution in login & role assignment process supports two operations: attributes value exchange and policy mapping. Two protocols are proposed to achieve privacy protection for the attributes value exchange. However, the second protocol is a complete ZERO knowledge disclosure protocol because RA in the first protocol may get to know the number of valid credentials AR holds. For policy mapping operation, we propose two protocols for mapping role assignment policies with attributes values. The first protocol is cryptography-based and handled by the AR. This protocol discloses some of RA's policies privacy such as which values are required for a role and the number of credentials the role requires, etc. The second protocol relies on a trust third party - Policy Verifier (PV). It is a complete privacy preserving solution for RA's attributes-to-role policies with ZERO policies knowledge disclosure.

The privacy protection solution for access request & response process needs to consider more on efficiency, system integration and compatibility. There are two types 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). Two sets of protocols are introduced for these two access requests. They are not ZERO knowledge disclosure protocols. However, without the involvement of complicated cryptography computation, they have greatly improved privacy protection than traditional SOA access control technology. Moreover, the solution is able to be implemented based on current SOA standards such as XACML, SOAP etc., and are practical and feasible to be implemented.

The rest of the paper is organized as follows. Section 2 discusses related work. In section 3, we propose a new RBAC scheme for SOA. Section 4 discuss privacy protection protocols for login & role assignment process. The privacy preserving protocol for access request & response are presented in section 5. Section 6 explain the security analysis of the proposed protocols. The implementation of the proposed protocols is explained in section 7. Finally, section 8 concludes the paper.

2. RELATED WORKS

T. Yu proposed "interoperable strategies for automated trust negotiation" [8]. He tried to protect sensitive credentials and services with access control policies and establishing trust incrementally through a sequence of credential disclosures. Complicated information exchange model are created to built trust negotiation. The disadvantages of his solution are obvious: it's not cryptography level solution and is easily for attacker to find security leak from the proposed

model. Also, it's not able to handle situation of dead cycling where both parties are not willing to first disclose their information. Furthermore, this solution is not manageable because different individuals have different standard to measure "sensitive" information. As a result, it's hard for their counterparts to handle different standards.

The Oblivious Signature-Based Envelope[9] approach proposed by W. Du is a cryptography-based solution, which perfectly solved the dead-cycling issue. The basic idea of this approach is that sender encrypts message in an "oblivious signature-based envelope" that receiver can open the envelope if he has the required certificate. The sender cannot know whether the receiver has the certificate. Oblivious Signature-Based Envelope is built on Identity-Base Encryption technology where sender encrypts sensitive information with receiver's ID as public key. Only receiver can decrypt it by private key from a trusted third party. This scheme is used to protect privacy for identity-based access control. It can be applied for a sender and a receiver, but cannot be directly used for the privacy protection of a distributed system, such as SOA.

K. Frikken introduced an attribute-based access control with hidden policies and credentials[10]. He discussed attribute and policy privacy protection in a more complicated access control environment. It provides a complete privacy-preserving solution for access requestor but cannot efficiently protect information provider's access policies.

J. Li et al. initiated an oblivious attribute certificates [11], which is actually a combination of the ideas proposed in [9] and [10]. It took use of similar technologies used in [10] to expand idea of [9] and it had the same shortcoming of [10]. Moreover, this approach is used for attributes-based access control rather than RBAC.

D. Yao et al. proposed a compact and anonymous role-based authorization chain[12], which discussed privacy protection in RBAC scenario. This paper provided privacy protection solution for the 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 is designed only to be applied in specific scenarios. In addition, it is inefficient because every delegation process needs a new one time key generation and signing permit.

3. ROLE-BASED ACCESS CONTROL IN SOA

The access process of SOA is comprised of two separate processes: login & role assignment and access request & response. Login & role assignment process is separated from typical SOA access control process. It is handled by a separate and independent sub-system. Access request & response process is still a part of traditional SOA structure.

Login is always the first step for all system's access. In our solution, clients or Access Requestors (AR) need to login 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 assigned to ARs are unified within the SOA range which are recognized and used by all individual systems.

Figure 1 provides an overview of login & role assignment process. Two independent third parties, RA and PV, are involved in the process. AR, a client or server requestor, obtains eligible roles through the interactions with RA and PV during the login process. RA works as an independent third party to maintain role-to-user assignment polices 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.
Certificate Authority (CA) provides attribute credential for AR. The CA verifies AR's attributes and issues credentials based on verified attributes. With the support from the Identity Based Encryption (IBE) technology, attributes credentials function as private key during the role assignment process.

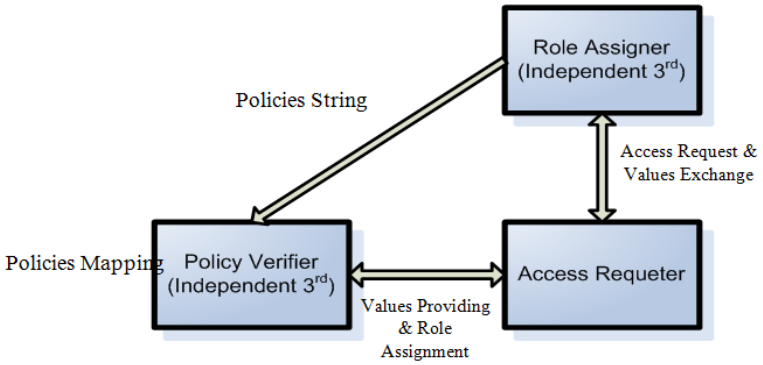

Figure 1: Login & Role Assignment Process

The login & role assignment process can be divided into the following three stages:

(1) AR sends access request to RA. AR and RA then are engaged in a value exchanged process (see section IV for details). AR will get a group of values representing the attributes credentials AR holds.
(2) AR then provides PV with the values he received from the first stage. RA provides PV with AR's value and policy string.
(3) PV will compare the values provided by AR and policies provided by RA, then PV will assign suitable roles to AR.

During the login & role assignment process, RA & PV know nothing about AR's attributes. AR and PV know nothing about RA's policies.

From the proposed RBAC scheme of the SOA model, we have privacy concerns for the following two processes:

- Login and role assignment
- Access request and response

The solution to protect privacy of the login and role assignment process will be explained in section IV. The solution to protect privacy of the access request and response process will be illustrated in section V.

4. PRIVACY PROTECTION PROTOCOLS IN VALUES EXCHANGE AND POLICY MAPPING

Three parties are involved in the login and role assignment process. They are AR, RA and PV. AR is the client of SOA. RA is an independent party above the composed units of SOA. RA maintains unified roles assignment policies that are agreed by all individual systems. AR is assigned roles by RA based on AR's attributes. In order to decide which role should be assigned

to AR, RA has to verify AR's attributes. However, AR may not willing to disclose his attributes to RA. That's the first privacy issue we need to solve. In addition, RA also needs to protect the privacy of his role assignment policies. That is, AR should not be able to guess or extrapolate RA's role assignment polices during the role assignment process. This is the second privacy protection issue we need to deal with.

Notations used in this section and the rest part of the paper are listed in the following table.

Name	Description
$ATTR_i$	Attributes i where $i \in \{1, n\}$. n is the number attributes
k_i	The *i*th attribute
$I(k_i, ATTR_i)$	Identity-based encryption on k_i using $ATTR_i$ as public key
$CRED_j$	Credentia for attribute j where $j \in \{1, m\}$ and m is the number of credentials
$I^{-1}(I(k_i, ATTR_i), CRED_j)$	Decryption of $I(k_i, ATTR_i)$. $CRED_j$ is the privacy key.
$SETINT(k_i[0], \{D_{i,1} \dots D_{i,m}\}, EA)$	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.
$Enc(Enc(Enc(RCred_s, k_1), k_2)\dots k_n$	Symmetric key encryption on $RCred_s$ n times using $k_1, k_2, \dots, k_n$ as key.
$Enc^{-1}(Enc^{-1}(Enc^{-1}(RCred_s, c_n), c_{n-1})\dots c_1)$	Decryption "$ERCred_s$" n times using $c_1, c_2, \dots, c_n$ as key.
$E_K(i)$	Identity-based encryption on i using K as public key. K is a required role.
$E_K(S)$	Identity-based encryption on "S" using K as public key. S is a random number.

Table 1: Notation

A. Solution One for Value Exchange

The first privacy preserving protocol is shown in Figure 2. Steps of the protocol are as follows:

1: For each attributes $ATTR_i$ required by all roles, RA created two random keys $k_i[0]$ & $k_i[1]$, and a public marker P. RA encrypts each $k_i[0]$ by $ATTR_i$ and send E_i (the i*th* attribute of RA) and P to AR. Now, for each $ATTR_i$, RA has a random key pair ($k_i[0]$, $k_i[1]$), the public marker P and E_i.

2: For each value α_i and AR's credentials $CRED_j$, AR generates $D_{i,j} = I^{-1}(E_i, CRED_j)$. If he holds m credentials, then he will get m D values for each E_i. AR then creates the homomorphic encryption system EA[13]. For each received E_i, AR gets a group values of the set ($D_{i,1} \dots D_{i,m}$), which are calculated by $D_{i,j}=I^{-1}(E_i, CRED_j)$.

3: AR and RA engage in the set intersection protocol[14], $SETINT(k_i[0], \{D_{i,1} \dots D_{i,m}\}, EA)$. AR's input are $\{D_{i,1} \dots D_{i,m}\}$ and EA. RA's input is $k_i[0]$. If there exists one value in the set $\{D_{i,1} \dots D_{i,m}\}$ equal to $k_i[0]$, RA knows EA[0]. Otherwise, it will be a random value.

For each attributes or Ei that RA sends to AR, RA gets $EA[x_i]$. x_i is 0 if AR has the credential. Otherwise, x_i will be a random number.

4: RA calculates $\delta_i = EA[x_i] * EA[k_i[1]] = EA[x_i + k_i[1]]$. He creates ordered pairs (δi, $Enc(P, k_i[1])$) and sends the pairs to AR. Also, for each attribute or value, RA calculates $Enc(P, k_i[1])$ and $\delta_i = EA[x_i] * EA[k_i[1]] = EA[x_i + k_i[1]]$ and sends them to AR.

5: When AR receives the pairs, he computers $\eta_j = DA(\delta_j)$ and $Dec(Enc(P, k_j[1]), \eta_j)$. If AR has the credential of $ATTR_j$, ($\eta_j = k_j[1]$), he gets P or he gets a random number. If AR gets P, he keeps $k_j[1]$, which prove that he holds a valid credential. AR will get P if he could successfully decrypt Ei by $D_{i,j} = \Gamma^{-1}(E_i, CRED_j)$ or AR cannot know P.

The privacy preserving protocol ensures that AR knows nothing about RA's policy and RA knows nothing about AR's credentials. However, AR knows the number of valid credentials he holds. To ensure that AR could not even know it, we introduce the second solution, which is a complete privacy preserving protocol.

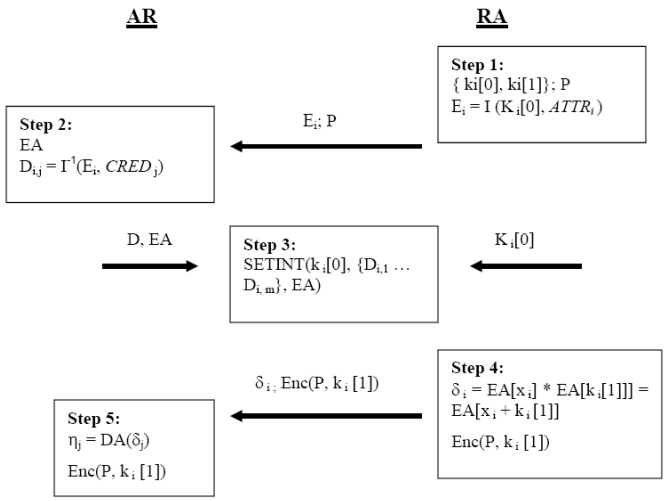

Figure 2: Values Exchange - Solution One

B. Solution Two for Value Exchange

The second privacy preserving protocol is shown in Figure 3. The steps of the protocol is as follows:

1: For each attributes ATTRi required by all roles, RA created two random keys $k_i[0]$ & $k_i[1]$. RA encrypts each $k_i[0]$ by $ATTR_i$ and send E_i (the i*th* attribute of RA) to AR.

2: AR creates a semantically secure homomorphic encryption system EA. For each value α_i and credentials $CRED_j$, he calculates $D_{i,j} = \Gamma^{-1}(E_i, CRED_j)$ for every E_i he received. If he holds m credentials, then he will get m D values for each E_i.

3: AR and RA engage 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. RA's input is $k_i[0]$. If one value in $\{D_{i,1} \dots D_{i,m}\}$ equals to $k_i[0]$, RA gets to know EA[0]. Otherwise, it will be a random value. For each attribute or E_i he sent to AR, RA gets the value of $EA[x_i]$ ($x_i = 0$ if AR has the credential or x_i will be a random number).

4: RA calculates $\delta_i = EA[x_i] * EA[k_i[1]]] = EA[x_i + k_i[1]]$ and sends δ_i to AR.

5: When AR receives the pairs, he computers $\eta_j = DA(\delta_j)$. Since there is no public marker P, he cannot know if η_j is $k_i[1]$ or not. For each required attributes, AR will get a value $k_i[1]$ if he

holds the credential for the attribute or a random value if he does not holds the credential. However, AR does not know if he gets a $k_i[1]$ or a random number.

6: For each attribute ATTRi, RA and AR engaged in a 1-out-of-2 OT protocol[15][16][17][18]. RA's input is the list of $\{r_i[0], Enc(r_i[1], k_i[1])\}$, and AR's input is η, which is $k_i[1]$ if he has the credential or a random value if he does not have it. If his input is $k_i[1]$, he decrypts $Enc(r_i[1], k_i[1])$, and gets $r_i[1]$ or he gets $r_i[0]$. AR sends list of $r_i[1]$ or $r_i[0]$ $i \in \{1, n\}$ to PV.

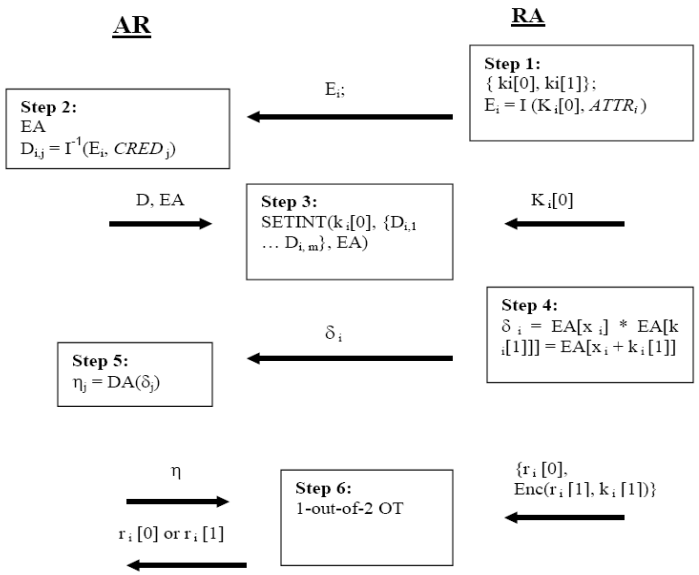

Figure 3: Values Exchange - Solution Two

C. PRIVACY PRESERVING PROTOCOL FOR POLICY MAPPING

After the credential verification process, AR has a bunch of values, some of them represent credentials he holds. These values will be mapping to RA's policies. Roles will be assigned to AR based on the mapping result. This step is handled by policy mapping methods that we will discuss in this section.

Solution One for Policy Mapping

If an AR matches all attributes of a role, he will be assigned credential of the role, RCreds. In RA's policy, there are *n* attributes required for the role, and there are *n* key values$\{k_1, k_2 \dots k_n\}$ accordingly. To get RCreds, RA must ensure that AR owns all these keys. Figure 4 provides us with steps of solution one for policy mapping. The protocol is shown as below:

1: RA picks up *n* random numbers $\{q_1, q_2, \dots q_n\}$. He encrypts the numbers using $\{k_1, k_2 \dots k_n\}$ as keys. Note that $Eq_j = Enc(q_j, k_j)$ and $j \in \{1, n\}$.

2: RA encrypts RCreds *n* times by $\{k_1, k_2 \dots k_n\}$ in the same order as he creates Eq_j. RA gets $RCreds = Enc(Enc(Enc(RCreds, k_1), k_2) \dots k_n)$. RA then pass $\{q_1, q_2, \dots q_n\}$, $\{Eq_1, Eq_2, \dots Eq_n\}$ and ERCreds to AR.

3: After AR receives the values, he calculates $Y=Enc^{-1}(Eq_n, c)$ where $c \in CC$. If Y equals to q_n, then $c= k_n$. AR keeps the value and decrypts other values of $Eq_{n-1} \dots Eq_1$. If AR decrypt the values, he gets all required keys $\{c_1, c_2 \dots c_n\}$ that are equal to $\{k_1, k_2 \dots k_n\}$.

4: AR applies the keys k_n, …k_2, k_1 to decrypt ERCreds and gets RCreds as role's credential Creds = Enc^{-1} (Enc^{-1} (Enc^{-1} (RCreds, k_n), k_{n-1})….k_1). If AR cannot get a value in {q_1, q_2,…q_n}, he cannot have all keys of the set {k_1, k_2 … k_n} and he is not qualified for the role's credential.

After this process, AR is able to know the number of valid attribute credentials he has, the amount of valid c values, the number of valid credentials RA requires for each role, and the values required for a specific role. Though RA has no idea of AR's credentials, the solution is still not a complete privacy preserving one.

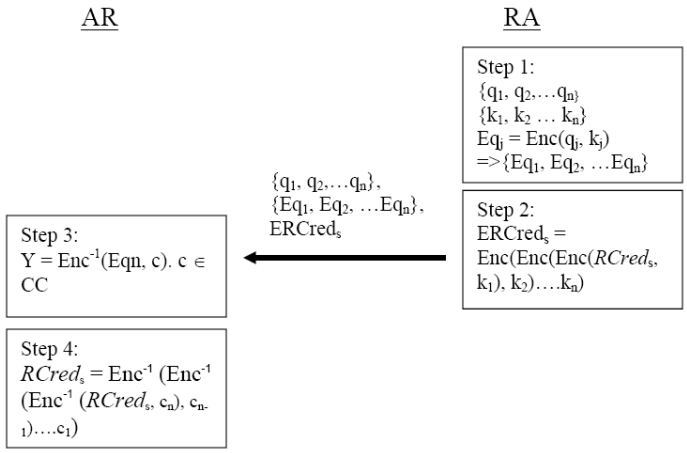

Figure 4: Policy Mapping - Solution one

Solution Two for Policy Mapping

To provide a complete privacy preserving scheme, a trusted third party PV is involved to handle the policy mapping task. In this approach, PV receives all policy lists from RA (p_1(k_{11}, k_{12},…$k_{1\,n1}$), p_2(k_{21}, k_{21},…$k_{2\,n2}$),…). PV gets all AR's key values set CC. PV will then map policies with AR's value with the following steps:

1. PV picks up one policy value set that is received from RA
2. PV checks if the role is constraint with other roles that are already assigned to AR.
3. PV checks if its parent role has already assigned to AR, if yes, assign this role to AR.
4. PV picks up one value in policy value set, and search the value from AR's value set. If finds a match, then picks up the next value in policy value set. If not found, return to step 1 to pick up another role's value set.
5. Repeat step 4 until the last value.
6. If all values are matched, then assign the role to AR.
7. Repeat step 1 until the last policy value set.

PV receives all policies that are represented by set of values from RA. PV has no knowledge about the values represents which attribute. As a result, PV knows nothing about RA's policies. Similarly, PV has no knowledge about AR's credentials either because PV does not know the relationship between the value and the credentials/attributes. The introduction of the third party PV has thus provided perfect privacy protection during the role mapping process.

5. PRIVACY PRESERVING PROTOCOLS FOR ACCESS REQUEST & RESPONSE

RB-XACML[18] is the standard to implement the core and hierarchical components of ANSI standard in SOA including roles, role hierarchies, permission-role assignment relation and user-role assignment relation. In RB-XACML, AR's role exists as an "attribute value". AR's attributes are transported between sub-systems or components. This is not a privacy friendly method because the AR may not want his role to be disclosed to SPs. We thus present the following protocols to provide the privacy protection of roles during the information access request process and service access request process.

A. Privacy Protection Protocol for Information Access Request

Role's privacy may not be as sensitive as personal attributes such as id, position, age and gender etc. However, it's desirable and valuable for the privacy protection of roles. In this paper, we make the privacy protection of roles as an option for AR who may choose his access request as "privacy mode". In most cases, AR & SP follow traditional SOA access control standards to realize access control. When "Privacy Mode" option is selected, SOA will follow the role privacy protection protocol for information access request illustrated in Figure 5.

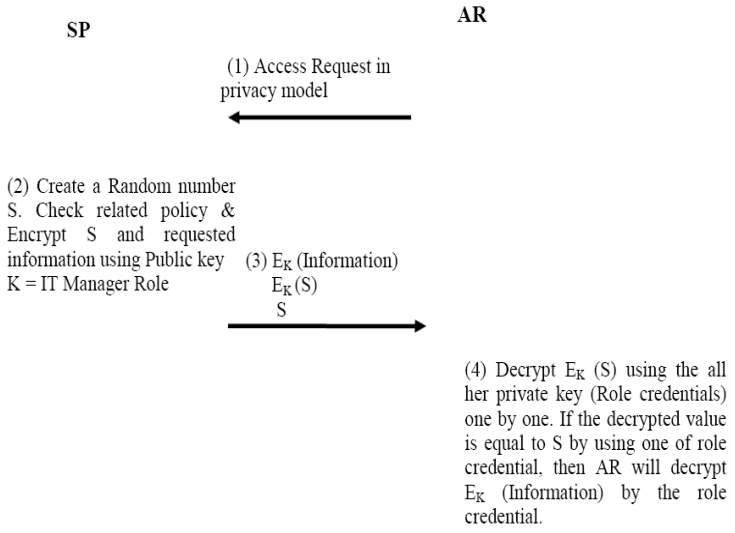

Figure 5: Privacy Protection Protocol for Information Access Request

The protocol is based on IBE technology. SP encrypts required information with role's title as public key, and sends the cipher text to AR. AR then decrypts the cipher text with her credentials obtained from the login and role assignment process. If AR holds private key of the role, he decrypts the cipher text. Detail steps of the protocol is as follows:

1. AR sends access request to SP. AR notifies SP if it's a role privacy preserving request or a normal request.
2. If it's a role privacy preserving request, SP checks his policies and search roles for the message. SP then creates a random number S, encrypt S and the message with the roles as public key.
3. SP sends the encrypted message and random number S and the number S to AR.

4. AR decrypts EK(S) by her role credentials as private keys. If there exits one role credential K_i such that $DR_i^{-1}(EK(S)) = S$, then the role credential of R_i is the requested role and AR can decrypt EK by R_i.

With the support of the protocol, SP knows nothing about AR's role. However, SP ensures that the AR can access the information if the AR holds required roles. Both AR's role privacy and SP's information security are protected.

B. Privacy Protection Protocol for Service Access Request

If AR requires to access a service such as a website, the situation will be difference. In this case, AR is requesting access to some resource of SP rather than some detailed data or text file which could be transferred.

Figure 6 illustrates the role privacy preserving protocol for service access request. The protocol is based on IBE technology as well. SP created a random number and encrypts the number with role's title as public key, and sends the encrypted result to AR. AR decrypts the cipher text with her credentials, which are obtained from the login and role assignment process. AR gets a set of decrypted number $\{S_1, S_2, \ldots S_n\}$. If AR holds private key of the role, he could get S though he does not know S and does not know if S is in her value set. The detail steps of the protocol is shown as below:

1. AR sends the access request to SP. AR notifies SP if it's a role privacy preserving request or a normal request.
2. If it's a role privacy preserving request, SP checks his policies to search roles for the message. SP then create a random number S, and encrypt the number using the roles as public key.
3. SP sends the encrypted S and EK (S) to AR.
4. AR decrypts EK(S) by her role credentials as private keys: $Si = DR_i^{-1}(EK(S))$ where R_i is the role credential, and gets a set of decrypted value set $\{S_1, S_2, \ldots S_n\}$.
5. He sends the value set to SP for further process.
6. After receives values set from AR, SP checks if there is a value S in the set. If yes, he grants access to AR or he will deny AR's access request.

In this protocol, AR has no knowledge about SP's policy. However, SP could know if AR's access permission is granted. That is, SP could 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 he has to grant permission by herself and know if AR has permission to access. SP just knows that AR holds one of required roles to access the service. He does not know which role AR has and also he has no knowledge about other roles held by AR.

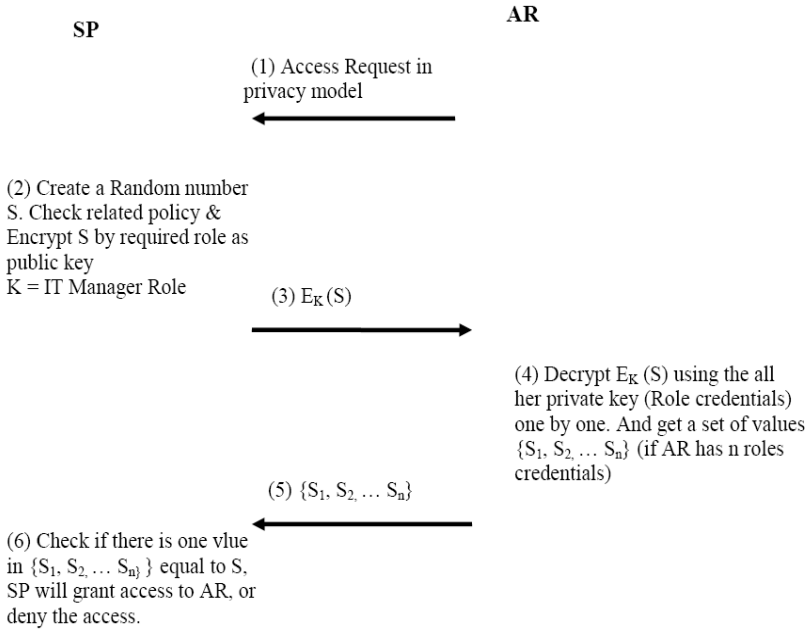

Figure 6: Privacy Protection Protocol for Service Access Request

6. SECURITY ANALYSIS

In this section, we take a brief review on our proposed protocols and see if they meet the following privacy protection targets:

1. AR's attributes privacy protection – ZERO disclosure of AR's attributes.
2. RA's policies privacy protection – ZERO disclosure of RA's policies.
3. All other privacy information protection – such as RA's role's credentials and AR's roles information.

A. Security Analysis for Login and Role Assignment Process

In login & role assignment process, RA encrypts values by attributes as public key using IBE. There is no non-neglectful possibility for AR to decrypt these values without credentials as private key. Otherwise, IBE will be unsustainable. AR cannot know these values if he does not hold valid credentials. In these two protocols, encrypted values are passed from RA to AR. RA receives nothing about AR's credential and as a result, RA has ZERO knowledge about AR's credentials and attributes. Since attributes in RA's policies are represented by random values in arbitrary order, AR is not possible to infer which value represents which attribute during AR's process of decrypting arbitrary ordered encrypted random values. In the first protocol, AR knows the number of credentials he has. If it is a privacy concern, we may choose the second protocol which is complete ZERO knowledge disclosure protocol.

In the policy mapping stage, we have two solutions. Since the first solution is not privacy-preserving, our analysis is thus focusing on the second solution. Since there is no direct information exchange between AR & RA in solution two, ZERO privacy disclosure are supported for both AR & RA. PV receives policies, values and encrypted role credentials from

RA. PV cannot decrypt role credentials without the shared key between AR and RA. Policy values are also meaningless to PV. Thus, PV has ZERO knowledge of RA's policy and role credentials. During the login and role assignment stage, AR sends PV a bunch of values, which are meaningless to PV. PV has ZERO knowledge about AR's attributes. After policy match, PV sends matched encrypted role credentials to AR. PV does not know the roles that AR gets.

B Security Analysis for Access Request & Response Protocols

Regarding to AR's role privacy, AR can set his role to be privacy protected. The proposed protocols will be employed if the AR chooses "Privacy Mode" or SOA will be executed with its traditional access process.

In the scenario of information access request, SP encrypts information with the requested role as public key and send the encrypted result to AR. SP has ZERO knowledge of AR's role because, if information is encrypted using IBE technology [9][10][20][21][22] by role's title as public key, only those who holds valid role credential as private key can decrypt it.

In the scenario of service access request, SP has to get involved to decide if permit should be granted to AR. As a result, SP knows if AR's request could be permitted. If the request is permitted, SP knows that AR holds at least one requested roles. This knowledge disclosure is not possible to avoid.

Regarding to SP's policy privacy issue, in the scenario of information access request, SP encrypts information by the requested role and sends the encrypted value to AR. AR decrypts the value by her role credentials. If AR is able to decrypt the random number and information with a role, he know this role is needed by the policy. This is still acceptable because we assume that SP's permission-to-role policies are not privacy sensitive. AR receives decrypted random number S, and decrypts it by all his role credentials and get a value set $\{S_1, S_2,\ldots S_n\}$. Since AR does not get involved in any polices related operations, he knows nothing about SP's polices.

In conclusion, we provide privacy protection option for AR, which protect AR's role privacy and SP's polices. Considering the factors such as system efficiency, compatibility etc., privacy protection in this process is not required to be "ZERO" knowledge disclosure. However, our protocols already make a great improvement comparing with the traditional RB_XACML standards.

7. IMPLEMENTATION OF OUR PROTOCOLS IN ACCESS REQUEST & RESPONSE

A. Implementation of Information Access Request Protocol

AR has the option to set her request as a normal request mode or privacy protection mode. If AR uses role privacy protection mode, its implementation is still compatible with SOA standards. We will show a sample RBAC policy and explain the implementation of those two options.

The sample RBAC policy we are using are as follows:

- AR holds a role of senior developer.
- AR's requested access is to access a document of "develper-guide.doc" .
- AR's requested action for the resource is "read".

```
<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 7: XACML Request in Tradition Way

```
  <Request>
bject>
   <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">
ttributeValue>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 8: XACML Request under Privacy Mode

Figure 7 and 8 show the implementation of the policy with the traditional SOA access request and the privacy protection mode. Suppose SP receives the request from AR, if the request is “Privacy Mode”, SP checks his policy set and find out which roles are eligible to read the information. In the privacy protection mode, role information is hidden. SP only knows the requested “resource” title. Thus, we need to create a set of resource PolicySets, which is in a similar format of Role PolicySets. In Figure 9 and 10, SP gets to know that role of “Senior Developer” will be granted read permission. The SP then encrypts the developer-guide.doc by the “Senior Developer” as public key. He also creates a random number N, and encrypts N by the “Senior Developer” as public key. SP then sends encrypted information and value N to AR. All values are transferred via SOAP, which is a file/data transfer standard for SOA and web service. Figure 11 is a sample of SOAP code used to transfer encrypted information, encrypted random number N and original random number N.

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

Figure 9: Resource PolicySet for developer

After AR receives the encrypted message, he decrypt “EncryptedNumber” by one of his role credentials. If the decrypted value matches “ChallengeNumber”, then use the credential to decrypt “EncryptedInformation” and get the information he needs. If it fails, he tries other role credentials. If AR does not own the required role credential, he will know nothing about the information.

Our protocol introduce new functions and algorithms to traditional SOA such as: the options for AR to setup “Privacy mode” 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.

```
<PolicySet PolicySetId="Permission PolicySet for developer-guide.doc"
        CombiningAlgorithm="permit-overrides">
  <Target>
        <Policy PolicyId="Permissions specifically for developer-duide.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 10: Permission PolicySet for developer

```
<?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>
        grIKlJMCSYHrgXlRThnxEYqZicqWeio0OJ3
        p+8NzFuqxzA8Yl55qaN/iy1Ywmm86fwqFmP
        8HL4/8lRA9dIfMySAkB5MF1KyEv5ReConcE
        DLoyl4sXJiYgWPQceh4XF06r49PkQGk8mvb
        WIpRbiiTJ76Uk22gCjdiU5IcWHnzB3k=
</ EncryptedInformation >
<! -- Encrypted Challenge Number N -->
        <EncryptedNumber>
        wDz/BvGUlJwL6WXNsc2/FGXiG9tlW4818VP
        wzlOSetiCSSz7kw4jwp1QvDJhJ+tr78X1uT
        zPkOQUbrUjHjaVnEwyP/Ez/uqVX7WW5zmvA
        y3ZtPmkkzHIJnM8f+FyRMG6Fr6nzZ/ZDEw6
        s+Vai5LTTLs3JZ297i5XTMAsaITgc74=
        </ EncryptedNumber>
<! - Original Challenge Number -->
        <ChallengeNumber>
        1234567890
        </ChallengeNumber>
       </ GetInformationResult>
    </ GetInformationResponse>
  </soap:Body>
</soap:Envelope>
```

Figure 11: Encrypted SOAP Message in Information Access Request

B. Implementation of Service Access Request Protocol

Implementation of service access request protocol is similar to that of information access request protocol. Especially the parts of access request, resource PolicySet, and permission PolicySet. Following the steps of the protocol, AR choose "Privacy Mode" and send SP the request, the request code is the same as what we used in the implementation of information access request. SP then checks resource PolicySet and permission PolicySet and creates a random value S. If the matched role is "Senior Developer", SP will encrypt S using "Senior Developer" as the public key. SP sends the encrypted S to AR via SOAP (See Figure 12). AR decrypts the encrypted S by all role credentials he holds. This process is handled at AR as an individual system or client, and it's not related with SOA. AR then sends the value set {S1, S2, … Sn} he gets to SP via SOAP (See Figure 13). AR reviews the received set $\{S_1, S_2, \ldots S_n\}$ and compare with the original number S to decide if granting or denying the AR's access.

```
<?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 12: Transfer Encrypted S with SOAP Message

The response code is exactly the same as traditional XACML response language code. The implementation of service access request can also take use of current XACML standards.

```
<?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 13: Transfer Encrypted {S1, S2, … Sn} with SOAP Message

8. CONCLUSION

In this paper, we provide privacy protection solutions for RBAC in SOA environment. We propose a new SOA RBAC solution which is privacy preserving. The RBAC solution is composed of two processes: login & role assignment 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 two protocols for each stage with different privacy protection and complication level. Access request & response process is regarded as less privacy sensitive and our solution is compatible with current SOA standards. In SOA, there are two kinds of access requests: information access and service access. We provide AR option to setup privacy mode and propose two protocols to handle privacy protection for these two types of access requests. We made a thorough security analysis on SOA RBAC solution and provide a set of new protocols, which make the whole process privacy friendly and are also beneficial for security policy management and system maintenance.

REFERENCES

[1] "Reference Model for Service Oriented Architecture 1.0", Committee specification 1, 2, August 2006.

[2] "SOA Security (red book)", IBM.

[3] J. Fiere, "SOA Security", Master Degree Thesis in Information Sciences, November 2007.

[4] N. A. Nordbotten, "XML and Web Services Security Standards", IEEE Communications Surveys & Tutorials, vol. 11. No. 3, 2009.

[5] "eXtensible Access Control Markup Language (XACML) Version 2.0, OASIS Standard, February 1, 2005.

[6] "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.

[7] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, C. E. Youman, "Role-Based Access Control Models", IEEE, 0018-9162/96, 1996.

[8] 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.

[9] N. Li, W. Du, D. Boneh, "Oblivious Signature-Based Envelope", Distributed Computing, vol. 17,no.4, pp.293-302, 2005.

[10] 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.

[11] J. Li, N. Li, "OACerts: Oblivious Attribute Certificates", IEEE Trans. On Dependable and Secure Computing, vol. 3, no. 4, October-December 2006.

[12] 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.

[13] Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/

[14] A. Menezes, P. van Oorschot, S. Vanstone, "Handbook of Applied Cryptography", CRC Press, 1996.

[15] B. Barak, "Oblivious Transfer (OT) and Private Information Retrieval (PIR)", November 29, 2007.

[16] R. Shahinian, T. Hu, "Foundations of Cryptography", March 16, 2005.

[17] W. Tzeng, "Efficient 1-Out-of-n Oblivious Transfer Schemes with Universally Usable Parameters", IEEE TRANSACTIONS. ON Computers, vol. 53, no. 2. February 2004.

[18] "Core and hierarchical role based access control (RBAC) profile of XACML v2.0, OASIS.

[19] D. Boneh, M. Franklin, "Identity-Based Encryption from the Weil Pairing", SIAM J. of Computing, vol. 32, No. 3, pp. 586-615, 2003.

[20] A. Shamir, "Identity-Based Cryptosystems and Signature Schemes", Advances in Cryptology – Proc. CRYPTO 1984, pp. 47-53, 1984.

[21] C. Cocks, "An Identity Based Encryption Scheme Based on Quadratic Residues", Proc. Eighth IMA International Conference Cryptography and Coding, pp. 360-363. December, 2001.

[22] M. Freeman, K. Nissim, B. Pinkas, "Efficient Private Matching and Set Intersection", Advances in Cryptology – Proc. EUROCRYPT 2004, pp. 1-19, May 2004.