Web Services Discovery in Secure Collaboration Environments

MOHAMED SHEHAB
Purdue University
KAMAL BHATTACHARYA
IBM T.J. Watson
and
ARIF GHAFOOR
Purdue University

Multidomain application environments where distributed domains interoperate with each other is a reality in Web-services-based infrastructures. Collaboration enables domains to effectively share resources; however, it introduces several security and privacy challenges. In this article, we use the current web service standards such as SOAP and UDDI to enable secure interoperability in a service-oriented mediator-free environment. We propose a multihop SOAP messaging protocol that enables domains to discover secure access paths to access roles in different domains. Then we propose a path authentication mechanism based on the encapsulation of SOAP messages and the SOAP-DISG standard. Furthermore, we provide a service discovery protocol that enables domains to discover service descriptions stored in private UDDI registries.

Categories and Subject Descriptors: D.4.6 [**Operating Systems**]: Security and Protection—*Access controls*; H.2.7 [**Database Management**]: Database Administration—*Security, integrity, and protection*; H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Web-based services*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Authentication*

General Terms: Security, Algorithms, Design

Additional Key Words and Phrases: Secure collaboration, secure access paths, encapsulated SOAP, private UDDI registries, protocols, services

The research of M. Shehab and A. Ghafoor was supported by the sponsors of the Center for Education and Research in Information Assurance and Security (CERIAS) at Purdue University, and by the National Science Foundation under NSF Grant IIS-0209111. During the summer of 2006, M. Shehab was supported by the IBM T.J. Watson Labs.
Authors' addresses: M. Shehab, University of North Carolina at Charlotte, Department of Software and Information Systems, 9201 University City Blvd., Charlotte, NC 28223; email: mshehab@uncc.edu; K. Bhattacharya, IBM T.J. Watson Research Center, Yorktown Heights, NY 10598; email: kamalb@ibm.com; A Ghafoor, School of Electrical and Computer Engineering, Purdue University, 465 Northwestern Ave. 305 N. University St., West Lafayette, IN 47907; email: ghafoor@ecn.purdue.edu.

ACM Reference Format:

Shehab, M., Bhattacharya, K., and Ghafoor, A. 2007. Web services discovery in secure collaboration environments. ACM Trans. Intern. Tech. 8, 1, Article 5 (November 2007), 22 pages. DOI = 10.1145/1294148.1294153 http://10.1145/1294148.1294153

1. INTRODUCTION

The vision of a global enterprise is quickly becoming reality. Enterprises of today are moving towards horizontal integration of business processes beyond national boundaries. Migrating processes across organizational boundaries has enabled companies to combine their efforts and become complex virtual enterprises [Afsarmanesh et al. 1998; Ludwig et al. 1999; Desai and Awad 2005; Ramnath and Landsbergen 2005]. The globally integrated enterprise leverages the collaboration among the company's different functions, operation components, and partners to enable the enterprise to excel.

Business operations that enable collaboration between the stakeholders associated to different business domains pose several security and privacy challenges. Consider business domains as an autonomous entity that provides a set of services and has its own administration and access control policies. Collaboration among multiple domains is at its core interoperation between the access control policies of the collaborating domains. Potential problems at this level have been analyzed and discussed in Gong and Qian [1996]. Secure interoperability in a multidomain environment is a challenging task [Gong and Qian 1996; Bonatti et al. 1997; Dawson et al. 2000], even in the presence of a trusted mediator managing security of such collaboration. In Shehab et al. [2005a, 2005b] we have proposed a framework for mediator-free secure collaboration that enables domains to cooperate in a dynamic and ad hoc manner with no trusted central mediator managing the collaboration environment.

Web service technologies and specifications enable the description, messaging, choreography, and composition of Web services [WSDL 2003; SOAP 2003; WSCI 2002; BPEL4WS 2002]. The WS-Security stack [WS-Security 2006] describes the different components that are deemed important to enable security for web services. In this article we consider a multidomain system where each domain provides a set of Web services. Our intent is to propose a mechanism to allow for both secure interoperability and service discovery in a mediator-free multidomain Web-service-based system. For example, in the healthcare system, our mechanism would enable physicians to discover and access services provided by other hospitals in different states. Figure 1 shows an example service collaboration between hospitals in different states. The edges between hospitals signify the established service collaborations. Using these collaboration agreements, physicians in Ohio (Hospital A) are able to acquire services in Minnesota (Hospital B). Using a combination of such collaborations enables physicians in Ohio to ultimately acquire services in California (Hospital D). When a patient from California moves to Ohio, our mechanism would enable doctors in Ohio to acquire authorizations to access services in California to acquire the patient's records stored in California. We use healthcare as a running example throughout this work. This article has three main contributions.

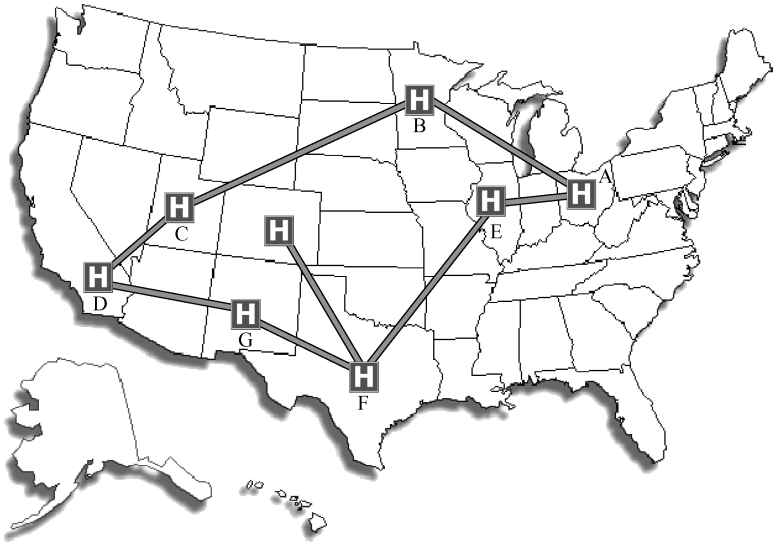

Fig. 1. Service collaboration between hospitals in different states.

First, we propose a multihop SOAP-based messaging protocol that allows domains to discover access paths to appropriate roles in different domains. In our multihop SOAP messaging protocol, a SOAP message encapsulates previous SOAP requests to enable domains to make access control decisions based on the request history. Second, we propose an authentication mechanism that ensures the integrity and authenticity of the encapsulated SOAP messages based on the SOAP-DISG standard [SOAP-DSIG 2001]. We show that the authentication mechanism is resilient to several types of attacks, such as path corruption, path replay, and colluding domains attacks. Third, we propose a service discovery protocol that enables domains to locate service descriptions (WSDLs) [WSDL 2003] stored in the private UDDI registries [UDDI 2003] of other domains. The service discovery protocol not only discovers services, but also finds authorizations required to access such services. We also present experimental results obtained from our implementation of the SOAP encapsulation technique on legacy Web servers.

The rest of the article is organized as follows. Section 2 describes some preliminary concepts to facilitate background for the rest of the work. Section 3 presents the proposed SOAP-based access path discovery protocol which is based on encapsulating SOAP messages. Section 4 presents the proposed service discovery protocol which enables domains in a mediator-free environment to publish and query private UDDI registries. In Section 5 we present our experimental results. Related work is discussed in Section 6. Finally, we present our conclusions in Section 7.

2. PRELIMINARIES

In this article, we assume a collaboration environment in which a role-based access control (RBAC) model [Ferraiolo et al. 2001, 2003] is adopted by all the

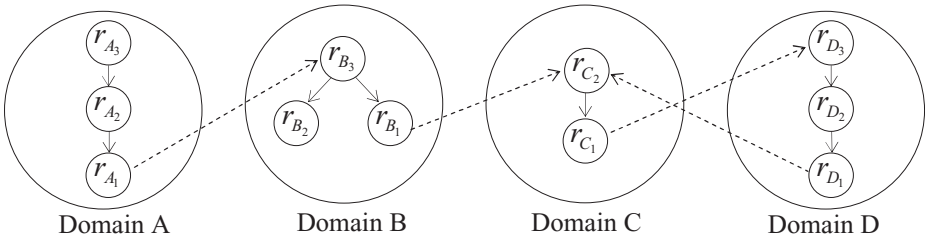

Fig. 2. Access paths and violations.

collaborating domains to model their access control policies. RBAC is suitable for specifying the security requirements of a wide range of commercial, medical, and government applications [Sandhu et al. 1996; Bertino et al. 1999; Atluri et al. 2001], and is being standardized by the National Institute of Standards and Technology (NIST) [RBAC 1996]. A domain that does not use RBAC as its access control model can easily generate an RBAC policy to join the collaboration. The analysis presented in the article can still be applied when other access control models are adopted. In RBAC, permissions are associated with roles. A set of permissions associated with a role is then granted to users by giving them membership of that role. The role permissions enable roles to execute a set of services where each role r is capable of executing a set of services referred to as the service set $SS(r) = \{s_1, \ldots, s_n\}$. The access control policy for domain D_i is modeled as a directed graph $G_i = \langle V_i, A_i \rangle$, where the vertex set V_i represents roles and the arcs set A_i portrays the dominance relationship between roles. For example, if role r_1 dominates r_2, ($r_2 \preceq r_1$), then $(r_1, r_2) \in A_i$. By using the RBAC permission inheritance properties [Crampton 2003] a user acquiring role r_1 can get permissions assigned to role r_2. For example, in a hospital the doctor role dominates the nurse role, giving a doctor the ability to acquire the access rights given to a nurse.

Collaboration among domains is achieved by introducing cross-domain mappings between roles in different domains. We will refer to such mappings as cross-links. A cross-link (r_x, r_y), indicates that user acquiring role r_x in domain $D(r_x)$ is able to acquire role r_y in domain $D(r_y)$. Figure 2 shows the cross-links as dotted edges connecting roles in neighboring domains. In this work, we assume that cross-links are selected by domain administrators according to the interoperability requirements of each domain. These links could be selected when the service-level agreements (SLAs) are negotiated [Myerson 2004; Dan et al. 2004]. A cross-link (r_x, r_y) starts at an *exit role* r_x and ends at an *entry role* r_y; we say that role r_y is an out-neighbor of role r_x and that role r_x is an in-neighbor of role r_y. For example, in Figure 2, domains A and B are neighbors. In cross-link (r_{A1}, r_{B3}), role r_{B3} is the out-neighbor of role r_{A1}. Assuming r_{A1} and r_{B3} are doctor roles in domains A and B, respectively, this would enable doctors in domain A to acquire services available for doctors in domain B. Furthermore, the domain administrators agree on a set of restricted cross-links that are prohibited to exist during the collaboration. These restricted access links are similar to the negative authorizations adopted in several access control models. Separation of duty constraints can be specified using a combination

of negative cross-links [Li et al. 2004] which we will refer to as the restricted access set, denoted by R. For example, in Figure 2 if $(r_{A1}, r_{D2}) \in R$, then a user acquiring role r_{A1} is not permitted to obtain role r_{D2}. In our healthcare example, negative authorizations enable hospitals to prohibit access from other hospitals which might be managed by competing healthcare providers.

2.1 Mediator-Free Secure Interoperability

According to Gong and Qian [1994, 1996] secure interoperability should ensure two principles. The first is the *principle of autonomy*, which requires that any access permitted within an individual domain must also be permitted under secure interoperability. The second is the *principle of security*, which requires that any access not permitted within an individual domain should not be permitted under secure interoperation. More formally, secure interoperability is defined as: Given n domains $G_i = \langle V_i, A_i \rangle$, $i = 1, \ldots, n$, a set of cross-links F, and a restricted access set R, an interoperation $Q = \langle \cup_{i=1}^{n} V_i, A_Q \rangle$, where A_Q is the resulting arc set $A_Q \subseteq \{\cup_{i=1}^{n} A_i \cup F\}$, is secure if it satisfies all of the following conditions:

(1) $A_Q \cap R = \emptyset$.
(2) $\forall u, v \in V_i, (u, v)$ is permitted in A_i if and only if (u, v) is permitted in A_Q.

Gong and Qian proposed several solutions to solve the secure interoperability problem using a central trusted third-party that has a global view of the all access control policies of the collaborating domains. In such a setting, all domains are required to share their access control policies with the central third-party and have to report any policy updates. The third party ensures that the principles of autonomy and security are not violated by selecting a subset of the established cross-links. Gong and Qian have shown that this problem is NP-complete, as well as that the compiled solution is static and has to be recomputed if any policy is changed or updated.

In our previous work [Shehab et al. 2005a, 2005b] we proposed a mediator-free framework to ensure secure interoperability between collaborating domains. A mediator-free collaboration is a completely distributed form of cooperation where domains comply in making access control decisions to avoid access violations. In a mediator-free environment none of the collaborating domains has a global view of all the access control policies; instead, the domains view the collaboration environment only through their established cross-links. In a mediator-free environment there is no central trusted third-party and domains are able to dynamically make access control decisions by utilizing the user's access history. The access history is the sequence of roles acquired from the home domain to the target domain in the collaboration environment, and we refer to it as the *access path*. For example, in Figure 2 the access path from role r_{A1} to role r_{D3} is $P = \{r_{A1}, r_{B3}, r_{B1}, r_{C2}, r_{C1}, r_{D3}\}$. Domains use the access paths to make access control decisions and to prevent security violations; this mechanism is similar to the Chinese wall security policy [Brewer and Nash 1989] where the access history controls future access control decisions. The access path enables domains to make localized access control decisions without

the need for a global view of the collaboration environment. Our framework translates the Gong and Qian [1994, 1996] secure collaboration requirements into secure path requirements to ensure secure collaboration. A *secure access path* is defined as follows.

Definition 2.1. Let $P = \{r_1, r_2, \ldots, r_n\}$ be an access path, where $i < j$ implies that role r_i was acquired before r_j. Let $D(r_i)$ denote the domain of role r_i. The path P is secure if it satisfies the following conditions:

C1. For all $i < j$ and $r_i, r_j \in P$, if $D(r_i) = D(r_j)$ then $r_j \preceq r_i$.
C2. For all $r_i, r_{i+1} \in P$, if $D(r_i) \neq D(r_{i+1})$ then $(r_i, r_{i+1}) \in F$.
C3. For all $i < j$ and $r_i, r_j \in P$, $(r_i, r_j) \notin R$.

Condition C1 ensures that roles acquired from the same domain are obtained according to the domain's role hierarchy. This ensures that the access control policies of the domains included in the path are not violated. Conditions C2 and C3 ensure that sets F and R are honored. When a cross-link is added to a secure path to ensure that the resulting path is secure, it has to be checked against the secure path conditions. We refer to these as the path linking rules [Shehab et al. 2005a]. In such a setting, domains are simply focused on ensuring that access paths are secure to avoid violations.

3. SOAP AND ACCESS PATH DISCOVERY

Simple object access protocol (SOAP) [SOAP 2003] is an XML-based messaging protocol. SOAP defines a set of rules for structuring messages that can be used for sending Web service requests and responses. It is not tied to any particular transport protocol, operating system, or programming language. SOAP messages consist of three parts, namely, an envelope, header data, and a message body. The envelope element identifies the XML document as a SOAP message. The header is an optional element which contains application-specific information such as authentication and payment information. The body element contains the actual SOAP message intended for the SOAP message receiver. In this section we propose a mechanism where SOAP messages are used to discover access paths between roles in different domains.

3.1 Access Path Discovery

Domains are able to collaborate with their neighbors via their established collaboration cross-links. Neighboring domains are single-hop collaborations which are inherently known by the involved domains. On the other hand, domains that have no established cross-links between them on which to collaborate need intermediary domains to establish secure access paths. To enable domains to discover the access paths available through intermediary domains, we propose using a SOAP-based distributed path discovery algorithm. For example, in Figure 1 path discovery would enable doctors in Ohio (Hospital A) to discover authorizations to acquire services in California (Hospital D). The path discovery protocol should ensure that discovered paths are secure by ascertaining whether they satisfy the secure path requirements discussed in the

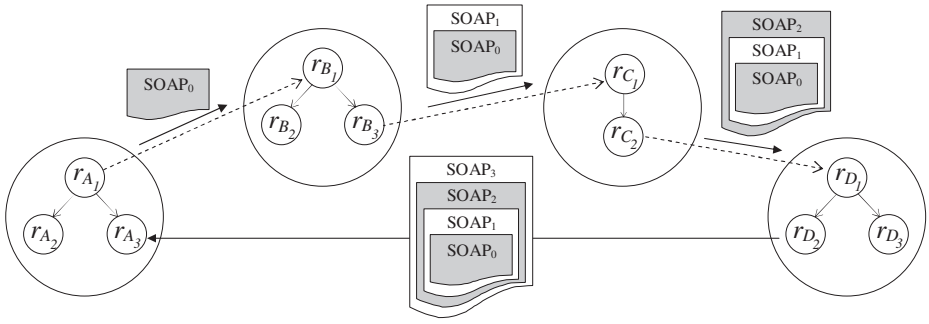

Fig. 3. SOAP message propagation.

preliminaries in Section 2. Furthermore, it should accumulate signatures that enable domains to verify the integrity and authenticity of the discovered paths.

When a home domain wishes to discover an access path to a role providing a certain Web service in a target domain, the home domain generates a path request SOAP message and sends it on its outgoing cross-links to neighboring domains. Upon receiving the path request message, the receiving domain extracts the accumulated access path from the received request. Then the extracted path is checked against the secure path rules to ensure that the path is secure according to the receiving domain's access control policy, the cross-link set, and the negative cross-links, as discussed in previous sections. The request is dropped if the path is not secure. Otherwise, if the receiving domain is the target of the received request, then it generates a reply and sends it to the requesting domain. If the receiving domain is not the target of the request, then it generates a new SOAP path request message and encapsulates the received request into the body of the new SOAP message. Then the new compiled SOAP message is forwarded to the domain's neighboring domains. Note that the complied SOAP message encapsulates the previous SOAP request, thus enabling domains to extract access path information, as will be discussed shortly. To avoid loops, requests are forwarded only to domains that are not already included in previously encapsulated requests. Furthermore, to control the size of the propagated access paths, a maximum path length which we refer to as $Pmax$ is enforced. Only paths having length below $Pmax$ are propagated. Figure 3 shows an example of an on-demand path discovery initiated by domain A from role r_{A_1} to determine access paths to roles reachable in domain D. Domain A generates a SOAP request ($SOAP_0$), sends it to its neighboring domain B. Domain B generates a new SOAP request message ($SOAP_1$), encapsulates $SOAP_0$ into $SOAP_1$, and forwards the request to domain C. Then domain C encapsulates $SOAP_1$ into a new SOAP request ($SOAP_2$) and forwards it to domain D. Using such a mechanism, the path information is accumulated by encapsulating previous SOAP requests with the current one; in this manner the domain receiving the request is able to extract the access path by examining the encapsulated requests. Referring back to the hospital example in Figure 1, SOAP message propagation would enable a doctor in Hospital A to discover roles in Hospital D.

```
<SOAP_i:Envelope>
  <SOAP_i:Header>
    <SOAP_i:Signature> ... </SOAP_i:Signature>
  </SOAP_i:Header>
  <SOAP_i:Body>
    <request>
      <currentRole> ... </currentRole>
      <nextRole> ... </nextRole>
      <targetRole> ... </targetRole>
      <targetService>
        <find_service>
          <name>
            Get Patient Records
          <name>
        </find_service>
      </targetService>
    </request>
    <SOAP_i-1:Envelope> ... </SOAP_i-1:Envelope>
  </SOAP_i:Body>
</SOAP_i:Envelope>
```

Fig. 4. SOAP path request message.

Figure 4 shows the format of the SOAP request message. The *currentRole* element is the role from which SOAP is being sent or forwarded. The $nextRole$ element is the role to which the request is being sent. The *targetRole* is the role that the request is targeting; the target role information is included only if the requester knows the target role. For example, a doctor in hospital A can request that the *targetRole* be a doctor in hospital D. The *targetService* element is a description of the target service that the request is querying: This component contains elements such as *find_business* and *find_service* which are commonly used to query web service description registries [UDDI 2003]. For example, a doctor might request the service "Get Patient Records" in a remote hospital. Note that SOAP message $SOAP_{i-1}$ is encapsulated in the body of SOAP message $SOAP_i$, the shaded portion in Figure 4.

By encapsulating SOAP messages and using the SOAP signature algorithms [SOAP-DSIG 2001], domains are able to accumulate the access path and to verify the integrity and authenticity of the collected access paths. The sequence of SOAP message encapsulation and forwarding is shown in the example described in Figure 5. Each domain updates the role information in the SOAP messages before forwarding it to its neighboring domains, thus enabling domains to extract the access path by simply traversing the encapsulating SOAP messages in the request.

3.2 Path Authentication

SOAP path request messages propagate between intermediate domains until the target domain is found. A mechanism is required to ensure the authenticity and integrity of the accumulated access path. The SOAP security extensions: digital signature (SOAP-DSIG) specification [SOAP-DSIG 2001] uses the XML signature specification [XML-Sig 2002] to represent an XML signature within a SOAP message. Using the SOAP message encapsulation mechanism proposed in the pervious section as well as the SOAP-DSIG specification, we propose a path authentication mechanism in this section. The proposed authentication mechanism is based on a signature that is generated by all the domains included

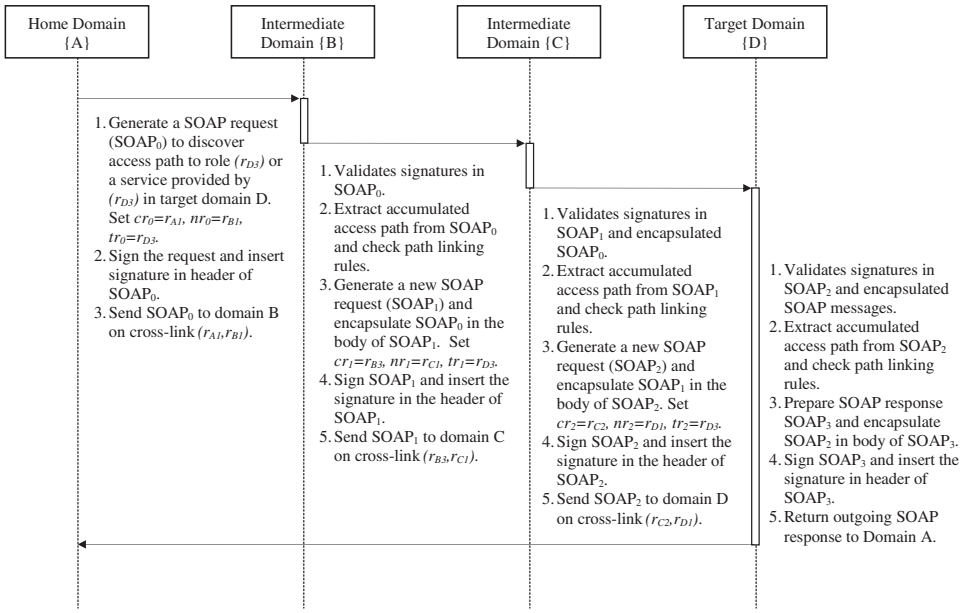

Fig. 5. SOAP intermediary access path discovery sequence.

in the access path. The authentication scheme should preserve both the path contents and path ordering.

A SOAP path request message $SOAP_i$ is composed of a body and header which contain the elements

$$\begin{aligned} SOAP_i &= \{Header_i, Body_i\} \\ Body_i &= \{cr_i, nr_i, tr_i, SOAP_{i-1}\} \\ Header_i &= \{Signature_i\}, \end{aligned}$$

where cr_i, nr_i, and tr_i are the *currentRole*, $next\,Role$, and *targetRole* in $SOAP_i$, respectively. Note that the body of $SOAP_i$ also includes the soap message $SOAP_{i-1}$. The soap signature is included in the header of the SOAP message and computed as

$$Signature_i = sign(h(Body_i), e_i),$$

where $h()$ is a secure one-way hash function [Schneier 1996], and $sign(M, K)$ is a signature function that signs the message M using the key K [Rivest et al. 1978]. Each domain D_i has a private key e_i and a public key d_i. Domain D_i computes the $Signature_i$ by computing the signing message digest of $Body_i$ using its private key e_i. The signature function has the property that $sign(sign(M, e_i), d_i) = M$. Note that this signature cannot be forged, as it is signed using the private keys of the involved domains. Figure 6 shows the SOAP request message ($SOAP_1$) sent from domain B to domain C in Figure 3. Lines (24–49) in the shaded region of Figure 6 indicate the encapsulated SOAP request message ($SOAP_0$) sent from domain A to domain B. Using the SOAP-DSIG standard, lines (06–07) indicate that the signature method used is based

```
(00) <SOAP-ENV:Envelope>
(01)   <SOAP-ENV:Header>
(02)     <SOAP-ENV:Signature
(03)       xmlns="http://schemas.xmlsoap.org/soap/security/2000-12">
(04)       <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
(05)         <ds:SignedInfo>
(06)           <ds:SignatureMethod
(07)             Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
(08)           <ds:Reference URI="#Body1">
(09)             <ds:DigestMethod
(10)               Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
(11)             <ds:DigestValue>P7lrN3f6rv33vKgbx··· </ds:DigestValue>
(12)           </ds:Reference>
(13)         </ds:SignedInfo>
(14)         <ds:SignatureValue>Lx9jje45KrsM3=··· </ds:SignatureValue>
(15)       </ds:Signature>
(16)     </SOAP-ENV:Signature>
(17)   </SOAP-ENV:Header>
(18)   <SOAP-ENV:Body SOAP-SEC:id="Body1">
(19)    <m:getAccessPath xmlns="http://www.domainC.com/ws">
(20)      <currentRole> DomainB.role3 </currentRole>
(21)      <nextRole> DomainC.role1 </nextRole>
(22)      <targetRole> DomainC.role1 </targetRole>
(23)    </m:getAccessPath>
(24)    <SOAP-ENV:Envelope>
(25)      <SOAP-ENV:Header>
(26)        <SOAP-ENV:Signature
(27)          xmlns="http://schemas.xmlsoap.org/soap/security/2000-12">
(28)          <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
(29)            <ds:SignedInfo>
(30)              <ds:SignatureMethod
(31)                Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
(32)              <ds:Reference URI="#Body0">
(33)                <ds:DigestMethod
(34)                  Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
(35)                <ds:DigestValue>j6lwx3rvEPOv··· </ds:DigestValue>
(36)              </ds:Reference>
(37)            </ds:SignedInfo>
(38)            <ds:SignatureValue>MC0CFFrVLtRlk=··· </ds:SignatureValue>
(39)          </ds:Signature>
(40)        </SOAP-ENV:Signature>
(41)      </SOAP-ENV:Header>
(42)      <SOAP-ENV:Body SOAP-SEC:id="Body0">
(43)        <m:getAccessPath xmlns="http://www.domainB.com/ws">
(44)          <currentRole> DomainA.role3 </currentRole>
(45)          <nextRole> DomainB.role1 </nextRole>
(46)          <targetRole> DomainC.role1 </targetRole>
(47)        </m:getAccessPath>
(48)      </SOAP-ENV:Body>
(49)    </SOAP-ENV:Envelope>
(50)   </SOAP-ENV:Body>
(51) </SOAP-ENV:Envelope>
```

Fig. 6. SOAP message encapsulation and signature.

on RSA. Line (08) specifies the part of the SOAP message for which the signature is to be computed, which refers to the `reference URI:"#Body1"` element. Note that line (18) shows that the SOAP body is given the same identifier `id="Body1"`, which indicates that the signature of $SOAP_1$ is computed by signing the digest of the body of $SOAP_1$.

When domain D_{i+1} receives SOAP message $SOAP_i$, the signature is verified by performing the check

$$h(Body_j) = sign(Signature_j, d_j) \quad \text{for all} \quad 0 \leq j \leq i.$$

Note that signature verification is performed using the public key information of the involved domains, thus the verification does not require contacting these

extractPath():
Input: msg = SOAP Message
Output: P = Extracted access path.

(1) If $verifySignature(msg) == false$ then Return $null$
(2) $P = \{msg.cr, msg.nr\}$
(3) While $(msg = getEncapsulated(msg))$ is not $null$
  (a) If $verifySignature(msg) == false$ then Return $null$
  (b) $P = \{msg.cr, msg.nr\}||P$
(4) Return P

verifySignature():
Input: msg = SOAP Message
Output: $true$ if signature is valid, $false$ otherwise.

(1) $i = domain(msg)$.
(2) $S = sign(msg.Signature, d_i)$.
(3) If $(S == h(msg.Body))$ then Return $true$ else Return $false$.

Fig. 7. Path extraction and signature verification algorithms.

domains. Figure 7 shows the detailed path extraction and signature verification algorithms.

3.3 Security Analysis

In this section we show the resiliency of our proposed SOAP authentication technique to several security attacks in a mediator-free collaboration environment. Throughout this section we assume that the sequence of SOAP requests is $\{SOAP_0, SOAP_1, \ldots, SOAP_{k-1}, SOAP_k \ldots, SOAP_n\}$, where, without loss of generality, assume that the SOAP message $SOAP_n$ is sent from domain D_i to domain D_{i+1}. The access path accumulated in SOAP message $SOAP_n$ is as follows:

$$P(SOAP_i) = \{cr_0, nr_0, cr_1, nr_1, \ldots, cr_{k-1}, nr_{k-1}, cr_k, nr_k, \ldots, cr_n, nr_n\}$$

3.3.1 *Path Corruption Attack.* The access path extracted from the SOAP path request message is one of the main components used for making access control decisions in a mediator-free environment. A malicious domain may attempt to alter the acquired access path by removing or adding entries to the current access path. The path corruption attack is composed of two types of attacks, namely, path insertion and deletion.

The *path insertion attack* is performed by an attacker in an attempt to insert a domain in the acquired access path. Given path $P(SOAP_n)$, the attacker attempts to alter the access path by inserting a SOAP message $SOAP_y$ in $SOAP_n$, which results in a new SOAP message sequence $\{SOAP_0, SOAP_1, \ldots, SOAP_{k-1}, SOAP_y, SOAP_k, \ldots, SOAP_n\}$, where the extracted access path is

$$\widetilde{P}(SOAP_n) = \{cr_0, nr_0, cr_1, nr_1, \ldots, cr_{k-1}, nr_{k-1}, cr_y, nr_y, cr_k, nr_k, \ldots, cr_n, nr_n\}.$$

The attacker is unable to generate the signature of the new SOAP sequence, as this requires the generation of new signatures for $SOAP_j$, $k \leq j \leq n$, which in turn requires knowledge of the secret keys e_j, for $k \leq j \leq n$. This indicates that the corrupted path cannot be authenticated by the attacker.

The *path deletion attack* is performed by an attacker in an attempt to delete a domain in the acquired access path. Given path $P(SOAP_n)$, the attacker attempts to alter the access path by deleting a SOAP message $SOAP_k$ in $SOAP_n$. This results in a new SOAP message sequence $\{SOAP_0, SOAP_1, \ldots, SOAP_{k-1}, SOAP_{k+1}, \ldots, SOAP_n\}$, where the extracted access path is

$$\widetilde{P}(SOAP_n) = \{cr_0, nr_0, cr_1, nr_1, \ldots, cr_{k-1}, nr_{k-1}, cr_{k+1}, nr_{k+1}, \ldots, cr_n, nr_n\}.$$

The attacker is unable to generate the signature of the new SOAP sequence, as this requires the generation of new signatures for $SOAP_j, k+1 \leq j \leq n$, and this requires knowledge of the secret keys e_j, for $k+1 \leq j \leq n$. Consequently, this path cannot be authenticated by the attacker. Note that other types of attacks such as path reordering are not possible because the attacker cannot prove the authenticity of such a path.

3.3.2 *Other Attacks.* In this subsection we discuss some other interesting attacks and propose some possible solutions to prevent and detect such attacks.

In a *path replay attack*, a malicious domain attempts to capture a request submitted during a valid session and tries to reply to it. This attack can be easily mitigated by extending the SOAP path request fragment to include a time interval in which the request is valid. The request fragment is in the SOAP body, which is signed by our proposed authentication technique and thus the attacker is not able to alter such information.

In a *colluding domains attack*, two or more domains in the collaboration environment collude to forge an access path. For example, two domains agree to provide a cross-link which did not previously exist. In this case, if the cross-link involves only the colluding domains, then this cannot be thought of as an attack, as both domains agreed to provide such a cross-link. However, if the cross-link involves domains other than the colluding ones, then the signature computed would require noncolluding domains, and this is similar to the path corruption attack discussed earlier. In this case the conspiring domains are not able to generate a fake signature.

The attacks discussed earlier are related to the authenticity of the access path. Other types of attacks, such as denial-of-service, eavesdropping, and masquerading, are assumed to be handled at other system layers. For example, eavesdropping can be handled at the transport layer by using HTTPS to transport packets between web servers.

4. SERVICE DISCOVERY IN MEDIATOR-FREE ENVIRONMENTS

Service discovery is an important component of Web service architectures and Web service composition [Medjahed et al. 2003]. Current Web service architectures [Schmidt et al. 2005; Cox and Kreger 2005] enable domains to publish service descriptions (WSDLs) [WSDL 2003] of their provided services in public universal description discovery and integration (UDDI) registries [UDDI 2003]. Services are simply discovered by querying the public UDDI registry. A mediator-free environment requires that no entity has a global view of the collaboration environment. In this section we propose a private UDDI infrastructure in which each domain maintains a private UDDI registry. Service discovery

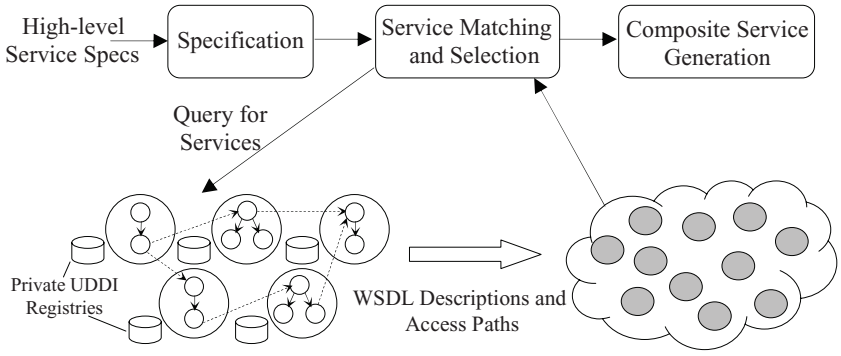

Fig. 8. Service discovery in private UDDI registries.

```
<xrta>
  <rta rta_id="RTA1">
    <role_name> Junior_Doctor </role_name>
    <serviceBag>
      <service_name> PatientRecordUpdate </service_name>
      <service_name> PrescriptionRecordCreate </service_name>
    </serviceBag>
  </rta>
  <rta rta_id="RTA2">
    <role_name> Senior_Nurse </role_name>
    <serviceBag>
      <service_name> PatientRecordRead </service_name>
    </serviceBag>
  </rta>
</xrta>
```

Fig. 9. Role-to-service assignments.

in our architecture intends to discover both services and the authorizations required to access such services. To enable both the discovery of authorizations and services, our service discovery is closely tied to the access path discovery introduced in previous sections. Figure 8 shows the proposed service discovery architecture as part of the general service composition framework [Medjahed et al. 2003]. In any business solution, it starts with high-level service specifications, and from there these specifications are mapped into a required set of composed services. Then the service discovery enables finding the necessary set of services to enable the composition of the required higher-level service.

4.1 Service Discovery Using Private UDDI Registries

In this architecture the UDDI registry is moved inside the domain; thus, for a domain D_i to access the UDDI registry of domain D_j, it should have sufficient authorizations. In a mediator-free environment authorizations are acquired by building access paths to roles in other domains. Each role in a domain is capable of performing a set of services that is dictated by the role-to-service assignment. In this architecture, to discover a service, domains should disclose paths to those roles that are able to perform such a service. The domain access control policy includes the access role hierarchy, the set of cross-links, the set of restricted links, and role-to-services assignments which map roles to services. Figure 9 shows the role-to-service XML document. Each role is assigned

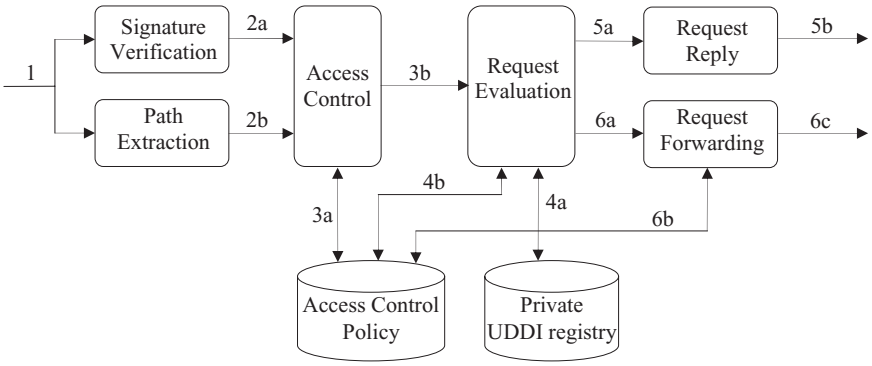

Fig. 10. Service discovery in private UDDI registries.

a *serviceBag* element which contains the names (identifiers) of services assigned to this role. For example, a *Junior_Doctor* role is assigned to services *PatientRecordUpdate* and *PrescriptionRecordCreate*.

Service discovery requests are sent using SOAP and follow the same SOAP encapsulation technique discussed earlier. When a home domain wishes to discover a service, the target role able to execute this service is not known, so the *targetRole* element in the SOAP path request message is not included (refer to Figure 4). On the other hand, the target service to be discovered is known and its description is entered in the *targetService* element in the request message, which can contain elements such as *find_business* and *find_service* that are commonly used to query UDDI registries [UDDI 2003] (please refer to Figure 4). The domain then forwards the request to its neighboring domains. When a domain receives a request its private UDDI registry is queried according to the entry role in that domain. If the domain is able to find the requested service, then it replies to the home domain. If the private UDDI registry does not contain the requested service, then the domain updates the access path by encapsulation and sends it to its neighboring domains. This enables domains to discover the requested services and the access path required for authorization of the requested service. Figure 10 shows the modules involved in the service discovery protocol in each domain. The numbered arrows in the figure represent different steps in our proposed discovery protocol.

Steps 1–2. When a service discovery arrives at a domain, the request signatures are verified to ensure its authenticity and the encapsulated access path is extracted. If the request passes the signature verification checks, it is passed to the access control module, otherwise the request is dropped.

Step 3. The access control module queries the access control policy to ensure that the accumulated access path is secure with respect to the path-linking rules, which check that the path does not violate the local hierarchy, and honors the set of cross-links and the set of negative links. If the access path is not secure with respect to the path-linking rules, the request is dropped. Note that the *nextRole* role in the service request SOAP message is a role in the current

domain, and the access control module assigns this role to the service request; we refer to this role as r_q. Assigning role r_q to the request indicates that the latter is able to discover services accessible to role r_q. The service request and assigned access role are forwarded to the request evaluation module.

Step 4. The request evaluation module queries the private UDDI registry using the information in the *targetService* element of the path request. It does so to discover services according to the service request criteria. The private UDDI returns a set of service identifiers; we refer to this set as S_c. Then the request evaluation module queries the access control policy to find the set of services assigned to the role r_q, which we refer to as the set S_{r_q}. Note that the set S_{r_q} includes all services assigned to roles that role r_q dominates. Then the request evaluation module computes the join between the sets of services S_c and S_{r_q}, which we refer to as the set $S_{result} = S_c \cap S_{r_q}$.

Steps 5–6. If the set S_{result} is empty, the request is updated and forwarded to neighboring domains. Otherwise, the domains send a reply to the querying domain indicating the set of services S_{result} and the set of encapsulated SOAP request messages to indicate the access path and path signature.

5. EXPERIMENTAL RESULTS

In this section we present experimental results generated from a proof-of-concept implementation of our SOAP-based path discovery protocol. All experiments were performed on an Intel Pentium IV CPU 3.2GHz with 512MB RAM, running Linux and Apache Tomcat. The SOAP encapsulation was implemented using the SOAP and the SOAP-DSIG standards using Java J2SE v5.0. Each domain was implemented as a separate Web server. The domain access hierarchy was implemented as a binary tree of access roles. The cross-links were generated a priori based on a neighborhood probability p, and these cross-links were then distributed on the involved domains. To evaluate the performance of the SOAP-based path discovery algorithm, each domain generated path requests and several performance metrics were recorded. The collected metrics for each path request include the discovered path length, the number of forwarded messages, the number of discovered domains, the number of discovered roles, and the number of replies received for each request. The collected statistics were averaged over all requests and all domains. In the following subsections we present the effects that several of the parameters, such as the number of domains, the maximum access path length $Pmax$, and the neighborhood probability p, had on the collected metrics.

5.1 Number of Domains

Several experiments were executed to investigate the effects of varying the number of domains on the collected metrics. In the following presented experiments the neighborhood probability p was set to 0.1. The maximum path length $Pmax$ was set to the number of domains in the collaboration environment; this removes the effect of $Pmax$, as its effect was studied in subsequent experiments. In our lab, we were able to set-up and run up to 26 domains running

our SOAP-based path discovery algorithm. Figure 11 shows the effects of varying the number of domains on different discovery metrics. The basic trend is that the different metrics increase as the number of domains in the system increase. For example, the number of discovered domains and roles increase with the number of domains (see Figures 11(b) and 11(c)). Note that the number of discovered domains and number of discovered roles follow a similar trend. Experiments were performed for different p values and they showed similar trends.

5.2 Effects of $Pmax$

The value of $Pmax$ controls the maximum allowable discovered path length. In this section the value of $Pmax$ was varied and metrics were collected for collaboration environments with 13 and 26 domains, respectively. The neighborhood probability p was set to 0.1. Figure 12 shows the generated results of such experiments where the value of $Pmax$ was varied from 4 in steps of 2. Note that by increasing the value of $Pmax$, domains are able to discern more roles because discovery requests are able to propagate farther into the collaboration environment. By our setting of $Pmax$ we are able to control the behavior of the discovery algorithm: For example, in Figure 12(d) we show that by setting $Pmax$ to 12 we were able to obtain average of an discovered roles in a 26-domain environment.

5.3 Effects of the Neighborhood Probability p

The neighborhood probability is the likelihood by which domains become neighbors. In the presented experiments we used 26 domains. The neighborhood probability p was varied and experiments were run for different values of $Pmax$. Figure 13 shows the experimental results. Note that as the number of neighbors increases, so does the number of messages forwarded and the number of replies per request, as indicated in Figures 13(a) and 13(b). Also note that the average discovered path length increases until $p = 0.2$, after which domains have more neighbors and thus shorter paths to requested roles; hence the average discovered path length decreases (see Figure 13(c)).

6. RELATED WORK

The problem of secure interoperation in a multidomain environment has been addressed in Cong and Qian [1994, 1996] and Shafiq et al. [2005]. In all such approaches a trusted third-party that has a global view of the collaboration environment is required to perform the security policy composition and integration. Dawson et al. [2000] presented a mediator-based approach to provide secure interoperability for heterogeneous databases. In this approach all access requests go through the central mediator which has a global view of the collaboration environment. Other strategies related to centralized database collaboration have been proposed in Morgensternet al. [1992], Jonscher and Dittrich [1994], Vimercati and Samarati [1997], and Wiederhold et al. [1998].

In the research community there has been an effort to highlight the challenges associated with Web service security. Research in the industry has had

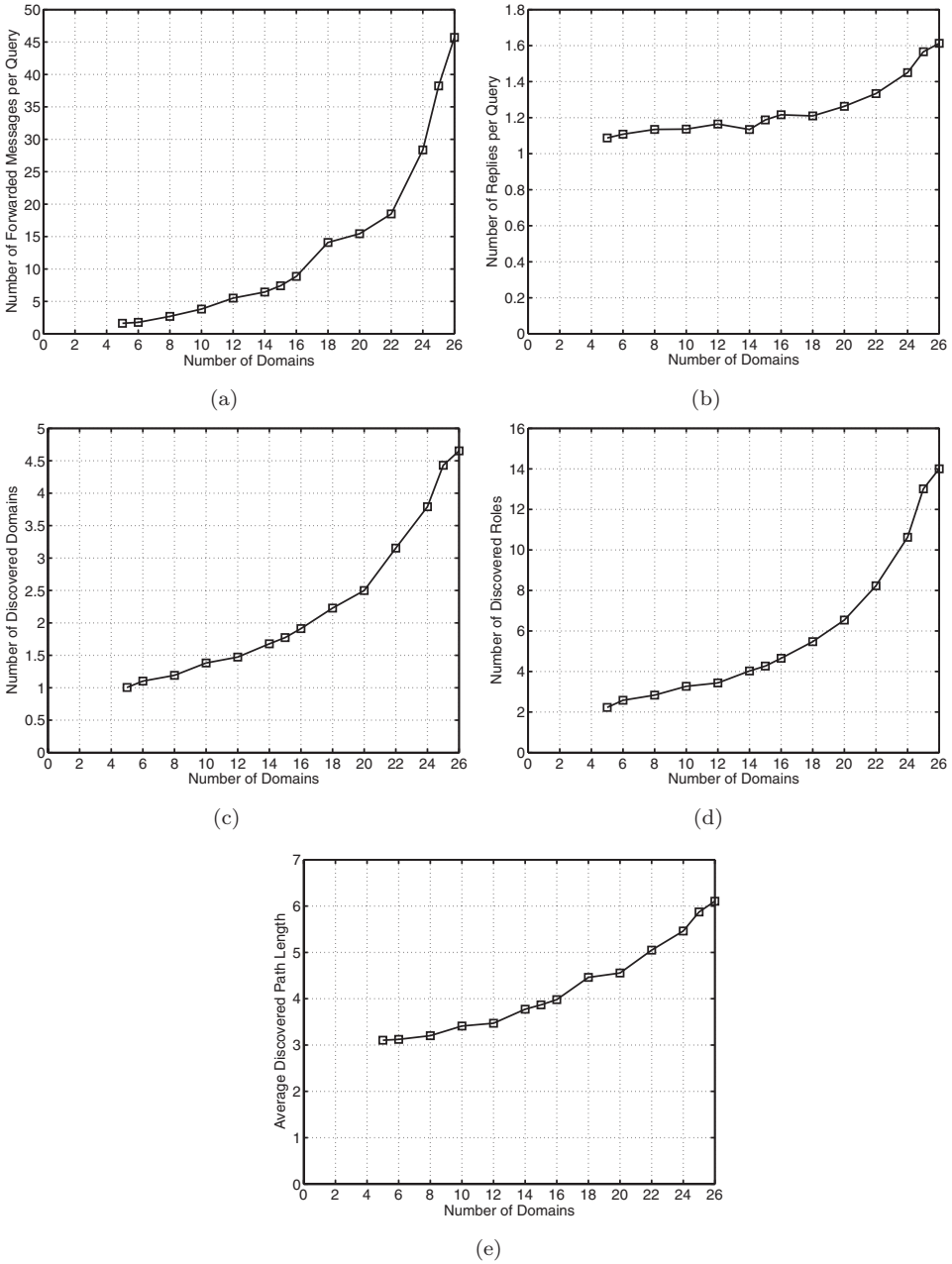

Fig. 11. Path discovery and number of domains.

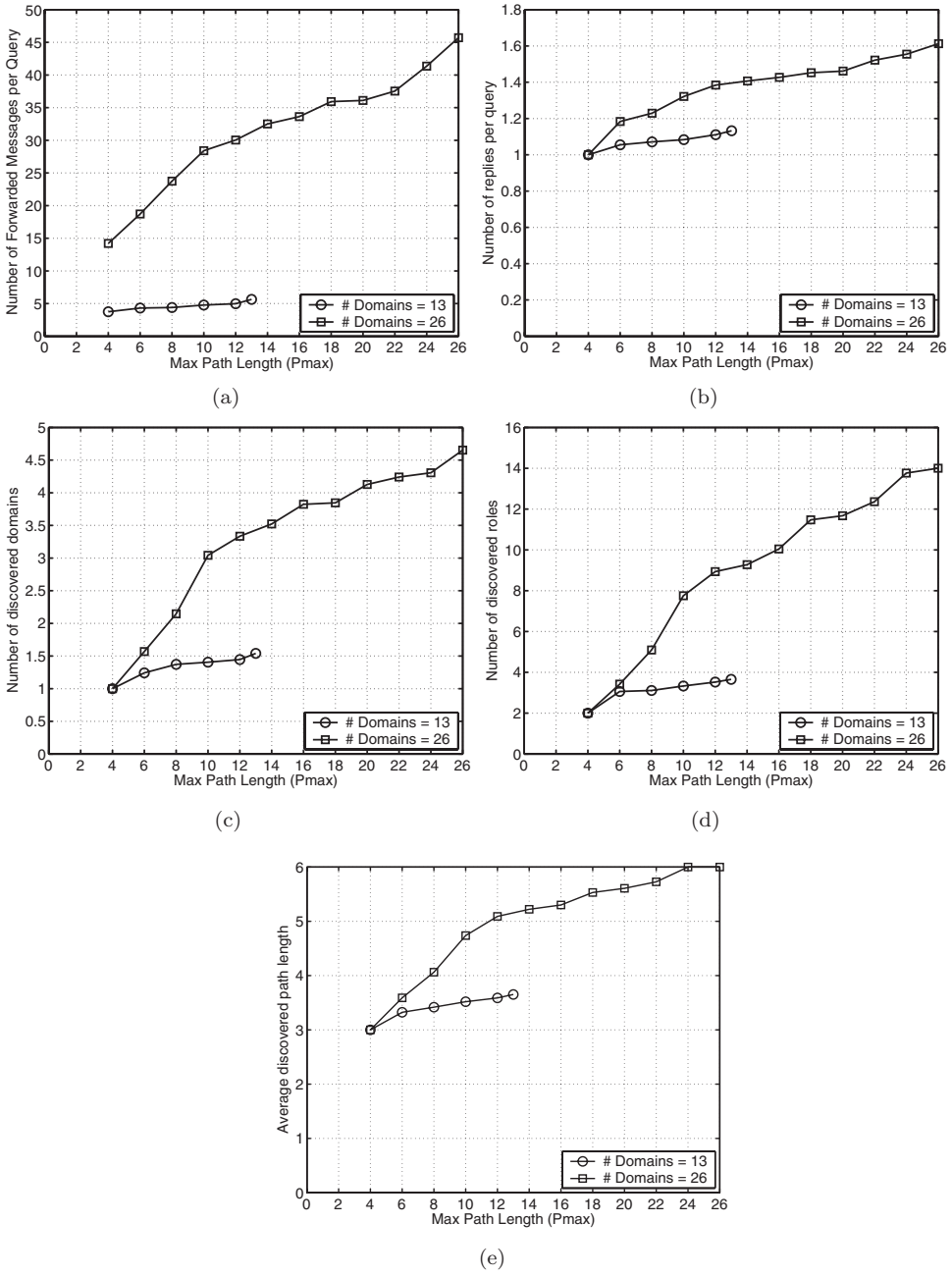

Fig. 12. Path discovery and maximum path length *Pmax*.

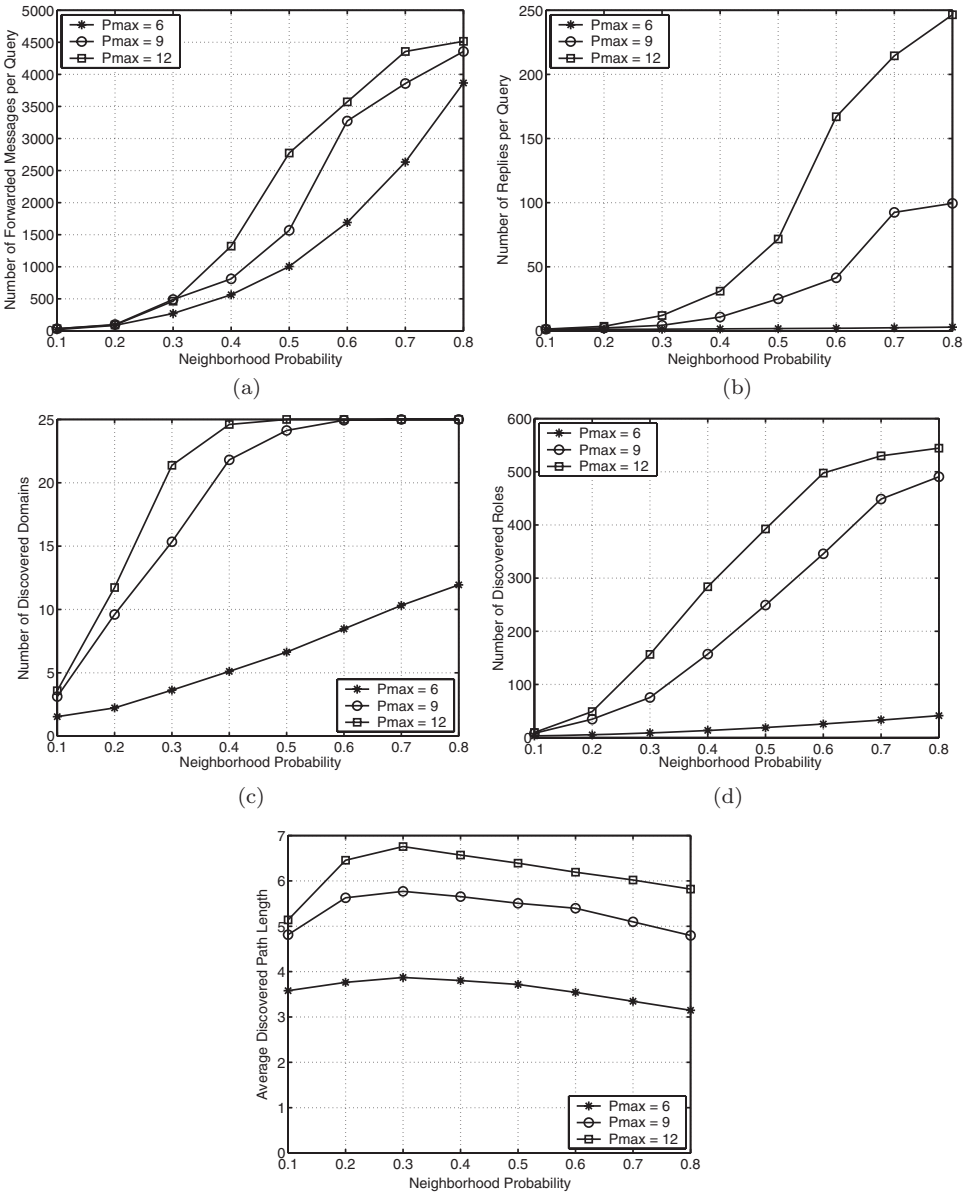

Fig. 13. Path discovery and neighborhood probability p.

a fair share of the related research in the area of Web service security, with standards such as security assertion markup language (SAML) [SAML 2004] and extensible access control markup language (XACML) [XACML 2005] having been recently adopted. SAML defines an XML framework for exchanging authentication and authorization information for securing Web services, and relies on third-party authorities for provision of assertions containing such information. However, SAML itself is not designed to provide support for managing access paths and collaboration policies; it is in fact a complementary specification. XACML is an XML framework for specifying context-aware access control policies for Web-based resources. XACML does not provide a formal mechanism to define cross-links and associations between different domains.

Another related set of emerging specifications consists of those outlined in the WS security roadmap [WS-Secmap 2002]. The roadmap consists of a number of component specifications, the main ones amongst these being WS-Security [WS-Security 2002], WS-Policy [2004], and WS-Trust [2004]. WS-Security is a specification for securing a whole or parts of an XML message using XML encryption and digital signature technology, and attaching security credentials thereto. WS-Policy is used to describe the security policies in terms of their characteristics and supported features. In fact, WS-Policy is a metalanguage which can be used to create various policy languages for different purposes, as well as to define an access control policy. Using the mechanisms provided by WS-Security and according to the policy requirements dictated by WS-Policy, the Web services trust (WS-Trust) defines a trust model that allows for the exchange of such security tokens in order to enable the issuance and dissemination of credentials within different trust domains, and to establish online trust relationships. In fact, WS-Trust could be used to generate the cross-links between different domains.

The models proposed in the roadmap have been directed primarily at the authentication aspect of Web service security, with an emphasis on designing secure messaging protocols to communicate security-relevant information such as security tokens and characteristics of security policy. The specification leaves room for the architecture to be augmented with custom authorization models. Our current work uses the SOAP messaging technique and SOAP-DSIG [SOAP-DSIG 2001] to implement the SOAP encapsulation mechanism to enable mediator-free secure collaboration. To the best of our knowledge, addressing this aspect of Web service access control is a novel contribution.

7. CONCLUSIONS

In this article, we have shown how to achieve mediator-free collaboration using the current Web service standards where domains collaborate to forward requests between themselves to enable the discovery of access paths and services. We first presented a secure path discovery protocol that enables domains to discover access paths to roles in other domains. The path discovery protocol is based on encapsulating and forwarding SOAP messages between collaborating domains. We presented a path authentication technique built on top of the SOAP encapsulation and SOAP-DSIG standard. We showed that the path

authentication technique is resilient to several path attacks. Next, we proposed a service discovery framework for mediator-free environments which is based on private UDDI registries. The framework enables domains to discover services in other domains and at the same time discovers authorizations to access such services. Finally, we presented experimental results obtained from our implementation of the presented SOAP encapsulation technique running on actual Web servers.

ACKNOWLEDGMENTS

The research of Mohamed Shehab and Arif Ghafoor has been supported by the sponsors of the Center for Education and Research in Information Assurance and Security (CERIAS) at Purdue University, and the National Science Foundation under NSF Grant IIS-0209111. During summer 2006, Mohamed Shehab was supported by the IBM T.J Watson Labs.

REFERENCES

AFSARMANESH, H., GARITA, C., AND HERTZBERGER, L. 1998. Virtual enterprises and federated information sharing. In *Proceedings of the International Conference on Database and Expert Systems Applications* (*DEXA*).

ATLURI, V., CHUN, S., AND MAZZOLENI, P. 2001. A Chinese wall security model for decentralized workflow systems. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS)*, ACM Press, New York, 48–57.

BERTINO, E., FERRARI, E., AND ATLURI, V. 1999. The specification and enforcement of authorization constraints in workflow management systems. *ACM Trans. Inf. Sys. Security 2*, 1 (Feb.), 65–104.

BONATTI, P., SAPINO, M., AND SUBRAHMANIAN, V. 1997. Merging heterogenous security orderings. *J. Comput. Secur. 5*, 1, 3–29.

BPEL4WS. 2002. Business process execution language for web services (BPEL4WS). http://www-128.ibm.com/developerworks/library/specification/ws-bpel/.

BREWER, D. AND NASH, M. 1989. The Chinese wall security policy. In *Proceedings of the IEEE Symposium on Security and Privacy*, 206–214.

COX, D. AND KREGER, H. 2005. Management of the service-oriented-architecture life cycle. *IBM Syst. J. 44*, 4.

CRAMPTON, J. 2003. On permissions, inheritance and role hierarchies. In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS)*, ACM Press, New York, 85–92.

FERRAIOLO, D., KUHN D., AND CHANDRAMOULI, R. 2003. *Role-Based Access Control*. Artech House.

DAN, A., DAVIS, D., KEARNEY, R., KING, R., KELLER, A., KUEBLER, D., LUDWIG, H., POLAN, M., SPREITZER, M., AND YOUSSEF, A. 2004. Web services on demand: WSLA-Driven automated management. *IBM Syst. J. 43*, 1 (Mar.), 136–158.

DAWSON, S., QIAN, S., AND SAMARATI, P. 2000. Providing security and interoperation of heterogeneous systems. *Distrib. Parallel Databases 8*, 1, 119–145.

DESAI, A. AND AWAD, N. 2005. Special issue on adaptive complex enterprises. *Commun. ACM 48*, 5 (May).

FERRAIOLO, D., SANDHU, R., GAVRILA, S., KUHN, D., AND CHANDRAMOULI, R. 2001. Proposed NIST standard for role-based access control. *ACM Trans. Inf. Sys. Security 4*, 3 (Aug.), 224–274.

GONG, L. AND QIAN, X. 1994. The complexity and composability of secure interoperation. In *Proceedings of the IEEE Symposium on Security and Privacy*, IEEE Computer Society, Washington, DC, 190–200.

GONG, L. AND QIAN, X. 1996. Computational issues in secure interoperation. *IEEE Trans. Softw. Eng. 22*, 1 (Jan.).

JONSCHER, D. AND DITTRICH, K. 1994. An approach for building secure database federations. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)*, Morgan Kaufmann, San Francisco, CA, 24–35.

LI, N., BIZRI, Z., AND TRIPUNITARA, M. 2004. On mutually exclusive roles and separation of duty. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*.

LUDWIG, H., BUSSLER, C., SHAN, M., AND GREFEN, P. 1999. Cross-Organisational workflow management and co-ordination WACC. *99 Workshop Rep. 20*, 1.

MEDJAHED, B., BOUGUETTAYA, A., AND ELMAGARMID, A. K. 2003. Composing web services on the semantic web. *VLDB J. 12*, 4 (Nov.), 333–351.

MORGENSTERN, M., LUNT, T., THURAISINGHAM, B., AND SPOONER, D. 1992. Security issues in federated database systems: Panel contributions. In *Results of the IFIP WG 11.3 Workshop on Database Security V*. North-Holland, 131–148.

MYERSON, J. 2004. Use SLAs in a web services context, part 1: Guarantee your web service with a SLA. http://www-128.ibm.com/developerworks/library/ws-sla/.

RAMNATH, R. AND LANDSBERGEN, D. 2005. IT-Enabled sense-and-respond strategies in complex public Organizations. *Commun. ACM 48*, 5 (May), 58–64.

RBAC. 1996. Role based access control (RBAC). http://csrc.nist.gov/rbac/.

RIVEST, R., SHAMIR, A., AND ADLEMAN, L. 1978. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM 21*, 2 (Feb.), 120–126.

SAML. 2004. Security assertions markup language (SAML). http://xml.coverpages.org/saml.html.

SANDHU, R., COYNE, E., FEINSTEIN, H., AND YOUMAN, C. 1996. Role-Based access control models. *IEEE Comput. 29*, 2 (Feb.), 38–47.

SCHMIDT, M., HUTCHISON, B., LAMBROS, P., AND PHIPPEN, R. 2005. The enterprise service bus: Making service-oriented architecture real. *IBM Syst. J. 44*, 4.

SCHNEIER, B. 1996. *Applied Cryptography*, 2nd ed. John Wiley.

SHAFIQ, B., JOSHI, J., BERTINO, E., AND GHAFOOR, A. 2005. Secure interoperation in a multidomain environment employing RBAC policies. *IEEE Trans. Knowl. Data Eng. 17*, 11, 1557–1577.

SHEHAB, M., BERTINO, E., AND GHAFOOR, A. 2005a. Secure collaboration in mediator-free environments. In *Proceedings of the 12th ACM Conference on Computer and Communications Security, (CCS)*, ACM Press, New York.

SHEHAB, M., BERTINO, E., AND GHAFOOR, A. 2005b. SERAT: Secure role mapping technique for decentralized secure interoperability. In *Proceedings of the ACM Symposium on Access Control Models and Technologies (SACMAT)*, ACM Press, New York,

SOAP. 2003. Simple object access protocol (SOAP). http://www.w3.org/TR/soap.

SOAP-DSIG. 2001. SOAP security extensions: Digital signature. http://www.w3.org/TR/SOAP-dsig.

UDDI. 2003. Universal description, discovery, and integration (UDDI). http://www.uddi.org.

VIMERCATI, S. AND SAMARATI, P. 1997. Authorization specification and enforcement in federated database systems. *J. Comput. Secur. 5*, 2, 155–188.

WIEDERHOLD, G., BILELLO, M., AND DONAHUE, C. 1998. Web implementation of a securtty mediator for medical databases. In *Proceedings of the IFIP 11th International Conference on Database Security*. Chapman and Hall, London, 60–72.

WS-POLICY. 2004. Web services policy framework (ws-policy). http://www-128.ibm.com/developerworks/webservices/library/specification/ws-polfram/.

WS-SECMAP. 2002. Security in a web services world: A proposed architecture and roadmap. http://www-128.ibm.com/developerworks/webservices/library/specification/ws-secmap/.

WS-SECURITY. 2002. Web services security (ws security). http://www-128.ibm.com/developerworks/webservices/library/specification/ws-secure/.

WS-SECURITY. 2006. OASIS web services security. http://www.oasis-open.org/committees/wss/.

WS-TRUST. 2004. Web services trust language (ws trust). http://www-128.ibm.com/developerworks/library/specification/ws-trust/.

WSCI. 2002. Web service choreography interface (wsci). http://www.w3.org/TR/wsci.

WSDL. 2003. Web services description language (wsdl). http://www.w3.org/TR/wsdl.

XACML. 2005. Extensible access control markup language (xacml). http://www.oasis-open.org/committees/xacml/.

XML-SIG. 2002. XML-Signature syntax and processing. http://www.w3.org/TR/xmldsig-core.

Received June 2006; revised January 2007; accepted April 2007