Volume 3, No. 6, June 2012
Journal of Global Research in Computer Science
ISSN-2229-371X
RESEARCH PAPER
Available Online at www.jgrcs.info

SOA BASED MULTI PARTY WEB SERVICES USING DYNAMIC AUTHENTICATION

Nalini Priya. G [#1], Balamurugan B [*2]
[#1] Associate Professor, Member IEEE, Department of Information Technology, KCG College of Technology, Anna University, Chennai, India, nalini.anbu@gmail.com
[*2] Asst.Professor (senior), School of Information Technology and Engineering, VIT University, Vellore, India, kadavulai@gmail.com

Abstract— Distributed applications has been a big boon for the development of several applications ranging from on-time supply chain management ,virtual collaborations and several kinds of service integration across organizations. Often this leads to new challenges in security and dependability. Collaborating services in a system with a Service-Oriented-Architecture (SOA) may belong to different security realms but often need to be engaged dynamically at runtime. If their security realms do not have a direct cross-realm authentication relationship, it is technically difficult to enable any secure collaboration between the services. A potential solution to this would be to locate intermediate realms at runtime, which serve as an authentication-path between the two separate realms. However, the process of generating an authentication path for two distributed services can be highly complicated. It could involve a large number of extra operations for credential conversion and require a long chain of invocations to intermediate services. In this paper, we address this problem by designing and implementing a new cross-realm authentication protocol for dynamic service interactions, based on the notion of service-oriented multi-party business sessions. Our protocol requires neither credential conversion nor establishment of any authentication path between the participating services in a business session.

Keywords: Authentication, Multi-Party Interactions, SOA, Secure Service Collaborations, Web Services, Workflow Business Sessions

INTRODUCTION

Dynamism and flexibility are becoming the core characteristics of modern large-scale distributed applications, such as business application integration, distributed auction services, and order processing [5][9]. A business process does not have to follow in many cases a strict business specification; the executing order of its activities is sometimes unpredictable, and on some occasions, the actual execution of a process can even be "one-of-a-kind" [7]. The applications and services involved in the process are typically heterogeneous and may be provided and maintained by different organisations. As an organisation has its own security mechanisms and policies to protect its local resources, the application across multiple organisations has to operate amongst multiple, heterogeneous security realms. A security realm is a group of principals (people, computers, services etc.) that are registered with a specified authentication authority and managed through a consistent set of security processes and policies.

Because organisations and services can join a collaborative process in a highly dynamic and flexible way, it cannot be expected that every two of the collaborating security realms always have a direct cross-realm authentication relationship. A possible solution to this problem is to locate some intermediate realms that serve as an authentication-path between the two separate realms that are to collaborate. However, the overhead of generating an authentication-path for two distributed realms is not trivial. The process could involve a large number of extra operations for credential conversion and require a long chain of invocations to intermediate services.

In this paper we present a new solution for dynamically authenticating the services from different realms. The main contributions of our work are: (1) using the multi-party session concept to structure dynamic business processes, (2) a simple but effective way to establish trust relationships between the members of a business session, and (3) a set of protocols for multi-party session management, supported by empirical evaluation and formal analysis. The following section 2 contains the background and related work done. In Section 3 we discuss the fundamentals of constructing multi-party service interactions. Section 4 describes our proposed authentication protocols and system with formal proofs.

BACKGROUND

The issues with cross-realm authentication have been discussed in many papers. For example, both direct cross-realm authentication and transitive cross realm authentication are supported in Kerberos [4] [17]. By using transitive cross-realm authentication, a principal can access the resources in a remote realm by traversing multiple intermediate realms, if there is no cross-realm key shared with the remote realm. However, Kerberos assumes that the authentication mechanisms in all the federated security realms are homogeneous. in practice, the authentication mechanisms in different security realms are often heterogeneous and even non-interoperable, both in structures and functions. in order to address the issue of federating such heterogeneous authentication mechanisms, credential conversion mechanisms are widely used in many existing solutions. The work in [12] presents two types of credential translator services, KCA which translates Kerberos credentials to PK credentials, and KCT which translates PK credentials to Kerberos credentials.

Reiter and Stubblebine in [16] argue that an authentication process in a large-scale distributed system often needs the

assistance of a path of security authorities as it is difficult to locate a single authority to authenticate all the principals in the system. They suggest using multiple paths to increase assurance on authentication. It is important to notice here that a Session Authority or SA in our system differs significantly from the security authority in [16]. A security authority is used to enforce security policies and processes for a security realm so as to prevent attacks from accessing the applications and resources within that realm. In contrast, an SA is associated with a business session (management system), independent of any local security realm. It has much simpler functionalities than a security authority, aiming to provide secure real information to session partners which may belong to different security realms.

The problems related to federation amongst heterogeneous authentication mechanisms used by different security realms are also discussed in the Web service federation protocol [1][10]. The Web service federation protocol defines a set of credential conversion mechanisms, with which a principal in a realm can convert its credential to a credential that can be accepted in another realm within the federation. The issues of discovering a credential chain is discussed extensively in [13]. It is shown that an authentication path can be found in polynomial time if there is a centralised entity which holds all the federation information of the security realms possibly involved. Considering that the session partners of a business session may be determined dynamically at runtime, it is practically difficult to have sufficient information about the security realms to be involved before the execution of that session. However, without such a centralised entity, this job becomes much more difficult. In the extreme case, all the realms possibly involved need to be searched before an authentication path can be identified. In order to realize peer-to-peer collaborations amongst Web services, IBM, Microsoft, and BEA proposed a specification, WS-Coordination [3], in August 2002. WS-Coordination describes an extensible framework for supporting the coordination of the actions in distributed applications. However, WSCoordination is intended only as a meta-specification governing the specifications of concrete forms of coordination. The security issues discussed in this paper are not addressed.

MULTI-PARTY SERVICE INTERACTIONS

In a distributed application, a *session* is a lasting collaboration involving several participating principals, called *session partners*. A session is often typified by a state which includes variables that hold information from messages transferred within the collaboration. A business process execution can be regarded conveniently as a *business session*. In terms of a Service-Oriented Architecture (SOA) [11], a business session is a collaboration involving two or more collaborative *services*, and has service *instances* as its session partners (a service instance is here referred to as a stateful execution of a service.) In practice, a session may discover and select services at runtime. After receiving an initial request from a business session, a service normally spawns a service instance to handle the request. Once this instance is accepted as a session partner, it is entitled to collaborate with other partners within the same session.

Although security to an extent is provided by the RFID system, which authenticates the incoming persons by means of their RFID tags some fraudulent may escape. This results in the hospital security system being endangered. In order to overcome such threats, unusual activities within the premises should be continuously monitored.

Two-Party Session:

As implied by the name, a two-party session consists of two session partners only, i.e. a client and a server. For the security of a two-party session, an authentication process is required when the client sends an initial request to the server. A short-term secret key between the session partners is then agreed upon and generated. The secret key, also called session key, can be used in further communications to encrypt the messages transferred between the session partners [8].

The two-party session technique is practically effective, and it is used widely in many distributed systems and integrated with the design of most authentication protocols (e.g. SSL and Kerberos [17]). However, new problems arise if the two-party session technique is applied directly to the construction of a multi-party session. Hada and Maruyama in [9] demonstrate that, if a multi-party session is constructed out of multiple two-party sessions, it is difficult in some cases for a session partner to verify whether the service instance it contacts is actually a member of the same session. From the perspective of cross-realm authentication, the two-party session technique does not address the issue with Heterogeneous Cross-Realm Authentication (HCRA), which requires credential conversion and the establishment of authentication paths.

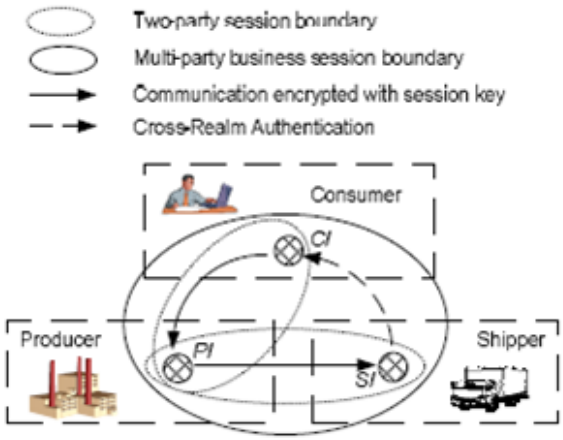


Figure. 1 A business session scenario

Figure 1 illustrates an example of a business session constructed with two two-party sessions. The business session consists of three participating services, *Consumer*, *Producer*, and *Shipper*. At the start of the business session, an instance of *Consumer*, *CI*, contacts *Producer* to order some products. After receiving the request from *CI*, *Producer* creates a service instance *PI* to handle it. *PI* then selects *Shipper* to deliver the products to *Consumer*. An instance of *Shipper*, *SI*, is thus generated to do this job, and it is required to negotiate with *CI* about delivery options and details. In this case, an HCRA process for authentication between *SI* and *CI* has to be performed by means of a new two party session as *SI* and *CI* do not know each other and

belong to different security realms. This HCRA process is both costly and complex due to credential conversion and possibly a long authentication path between the two local authentication systems of *SI* and *CI*. For a business session involved with n heterogeneous security realms, the HCRA process has to be repeated $n \times (n - 1)/2$ times to allow all possible partner interactions with the session.

Multi-Party Session:

A multi-party session may have two or more session partners for the intended collaboration. A partner can search for and invoke new services at runtime. Before a service (instance) is accepted as a new partner, an HCRA process is needed. However, unlike a two-party session, authentication for the existing partners of a multi-party session could be simplified significantly without requiring credential conversion and the establishment of any authentication path. This is because session partners can make use of their session memberships to authenticate each other even if they belong to different security realms. A shared session key or individual secret keys may be used to enforce a secure collaboration amongst session partners. Consider the example of Fig 1 again. When *SI* attempts to contact *CI*, it does not have to authenticate itself with the local authentication system of *CI* because both *SI* and *CI* are members of the same session. *SI* can simply use its session membership to prove its identity to *CI*. This simplified authentication process is called Simplified Cross-Realm Authentication (SCRA). The HCRA process has to be repeated $(n - 1)$ times for a multi-party session with n security realms, but up to $(n - 1) \times (n - 2)/2$ authentication processes can be simplified as SCRA based on session memberships, thereby reducing both cost and complexity significantly. However, managing and coordinating a multi-party session is more complex in nature, in comparison with handling two parties only. A multi-party session management system needs to address the issues with *message routing* and *secret keys for communications*. A *Session Authority* (SA) is also required to provide reliable real-time information (e.g. memberships) about session partners [9].

Message Routing:

Message routing is concerned with the issues of dispatching messages to the intended service instance which maintains corresponding states. In practice, a service may handle requests from different requestors concurrently. When all the requestors invoke operations provided by the same port, the messages are sent to the same address (e.g. the same URL). In this case, additional correlated information is needed, which helps the underlying middleware to determine which interaction a message is related to and to locate the corresponding service implementation object to handle the message.

A simple approach is to exploit a correlated token, shared by the communicating partners, for identifying the related messages transported within the collaboration. A shared token is sufficient to the identification of session partners on the both sides of two-party collaboration. However, session partners (i.e. service instances) in a multi-party session may be generated by the same service with the same address. It is difficult to distinguish them using a single token. In contrast with the token-based solution, an ID-based solution assigns every session partner with a unique identifier, thereby distinguishing all the partners unambiguously. In practice, a token-based solution is usually used to decide whether an instance is actually working within a business session while an ID-based scheme is employed to identify individual session partners in the case that fine-grained instance identification is needed.

Secret Keys:

In a two-party session, authentication typically consists of several rounds of operations and message passing, and the session key used in the subsequent communication between the two partners is normally a by-product of the authentication process. However, in a multi-party session, SCRA is a highly simplified process and does not include the automatic generation of secret keys.

An obvious approach is to generate a single secret key for a given multi-party session and then distribute it to all the session partners. Once the session key is generated, it can be used to simplify the authentication process amongst the existing session partners, thereby avoiding HCRA. Hada and Maruyama's protocols in [9] are an example of this type of solution with the support of a Session Authority. However, if a partner loses the secret key, the security of the whole session will be compromised. Moreover, session partners may leave and join a session dynamically. When a partner leaves from its session, the shared secret key must be refreshed in order to ensure that any previous partner cannot gain any further information from the session. Similarly, when a new partner joins the session, the secret key must also be refreshed in order to ensure that any new partner cannot obtain any previous information transferred within the session. The issues related with secret key revocation have been discussed in many papers on secure group communications (e.g. [15][20]).

Another possible solution is to generate a shared secret key for every pair of session partners (e.g. using the Diffie-Hellman public key algorithm [18]). This scheme is more costly but it avoids the issue with key revocation.

Session Authority:

A Session Authority (SA) is a service that provides reliable real-time information (e.g. session memberships) for a given multi-party session. For example, the SA may be employed to notify that a partner has left from the session, by contacting all the partners that have collaborated with the previous partner. An SA service could be associated conveniently with, or implemented as part of, a multiparty management system. This can be implemented using different methods with different features and characteristics such as fault-tolerance, scalability and cost-effectiveness. These methods include centralized management, decentralized architecture for better scalability, and fully distributed information provision for improved fault-tolerance. As an example of the SA implementation, our authentication protocols are designed to conform to the WS-Coordination specification [3] in which an SA is an extension of a *coordinator*. In WS-Coordination both centralized and decentralized coordinators are discussed. An SA may act as a centralized service that handles requests from all the session partners within a business session; alternatively, an SA may manage the session partners within

a local domain only, and a group of decentralized SA's can then manage collectively the whole business session, thereby avoiding the problem of concentrating the SA operations in a single place.

AUTHENTICATION PROTOCOLS

In this section we provide a multi-party authentication system and use the business scenario in Section 2 to explain the structure of the system. The related protocols are described and analyzed formally.

Example:

Consider an SA-based multi-party authentication system. In this system each business session is associated with a unique session identifier. Every service instance within a session is associated with a unique instance identifier so that every session partner can be identified unambiguously. The Diffie-Hellman public key algorithm is used to generate a pair of public/private keys for each service instance. The public key of an instance is identical to its identifier and can be transferred over the network while its private key is kept securely and can be used to prove the possession of the identifier. The Diffie-Hellman algorithm is also exploited for generating a shared secret key for every pair of collaborative partners of a session.

Figure 2 illustrates how the authentication system performs multi-party session authentication and management using the example of Figure 1. First, *CI* contacts an SA to start a new business session, *S*. The SA service then generates an instance, *SA*, to manage the new session. *CI* thus becomes a session partner of *S*, and its identifier is recorded in *SA*. *CI* then contacts *Producer*. *Producer* sends back the identifier of the instance *PI* in Step (2) while *PI* is introduced by *CI* to *SA* in Step (3). Next, *CI* starts to collaborate with *PI* after receiving the confirmation from *SA* (Step (4)). In the same way, *PI* invokes a new shipper instance *SI* and introduces it to *SA* (Steps (5) to (7)). After receiving the request from *SI*, *CI* first contacts *SA* to check whether *SI* is a legal business session partner of *S* (Steps (8) and (9)). Once this is confirmed by *SA*, *CI*

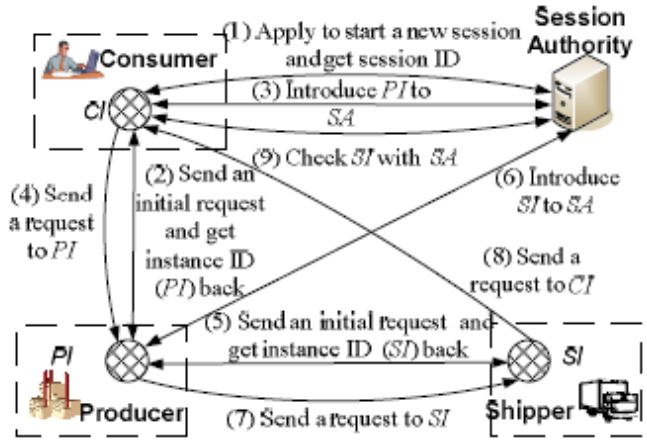


Figure. 2. A business scenario

Formal Definitions:

In this section we will define two core protocols in our multi-party authentication system using the well known Logic of Authentication (or BAN logic) [2]. Protocol 1 is concerned with the introduction of a new session partner, and Protocol 2 performs authentication between two existing session partners. For the brevity of discussion, we use the following notation for formal definitions and proofs (which is a simplified version of the notation used in [14]).

p	large prime number
a	exponentiation base
A, B, C	session partners
SA	session authority
ID_A	identifier of A
S	multi-party session with identifier ID_S
$Pri(A)$	private key of principal A
$Pub(A)$	public key of principal A, i.e. $(a^{Pri(A)} \bmod p) = ID_A$
$K_{(A,B)}$	secret key generated with $Pri(A)$ and $Pub(B)$; $K_{(A,B)} = (Pub(B))^{Pri(A)} = a^{Pri(A)\,Pri(B)} \bmod p$; $K_{(A,B)} = K_{(B,A)}$
(M, N)	composite message composed of messages M and N
$MAC(M)_K$	message authentication code of M generated with secret key K
$Secure(M)$	message M is transmitted by a secure channel
$Valid(M)_K$	composite message $(M, MAC(M)_K)$
$\uparrow Pub(A)$	$Pub(A)$ is *good* [14], and its corresponding $Pri(A)$ will never be discovered by any other principals
$\#M$	M is fresh, i.e. M has not been sent in a message at any time before the current run of the protocol
$SP(A, S)$	statement that A is a session partner of S. Particularly, $SP(SA, S)$ is always true
$A \xleftarrow{K'_{(A,B)}} B$	$K_{(A,B)}$ is A's secret key to be shared with B, but not yet confirmed by B
$A \xleftrightarrow{K_{(A,B)}} B$	$K_{(A,B)}$ is A's shared secret key and confirmed by B
$A \equiv X$	A believes that statement X is true
A says X	A sent a message including statement X
$A \Rightarrow X$	A is an authority on X, i.e. A has jurisdiction over X
$A \lhd M$	A receives message M

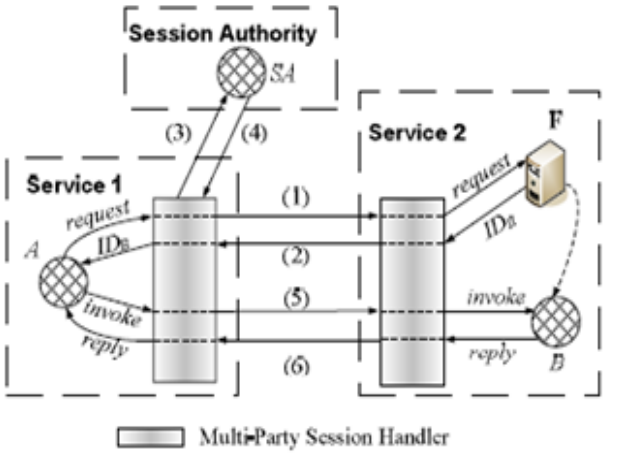


Figure.3 Protocol 1: Accepting a new session partner

Figure 3 illustrates Protocol 1: Accepting a new session partner. Our protocol conforms to the WSResource Framework (WSRF) specification [6], where a service is associated with a factory service *F* that generates service instances.

The details of the messages transported within Figure 3 are presented as follows, where "$A \rightarrow B$" means that *A* sends a message to *B*:

(1) $A \rightarrow F$: $Secure(Request, ID_S, ID_A)$
(2) $F \rightarrow A$: $Secure(ID_B, ID_S)$
(3) $A \rightarrow SA$: $Valid(SP(B,S), ID_B, ID_A, ID_{SA}, ID_S, N)_{K(A,SA)}$
(4) $SA \rightarrow A$: $Valid(Confirm, N+1)_{K(SA,A)}$
(5) $A \rightarrow B$: $Valid(invoke, ID_A, ID_B, ID_S, N_1)_{K(A,B)}$
(6) $B \rightarrow A$: $Valid(reply, ID_B, ID_A, ID_S, N_1+1)_{K(B,A)}$
where N and N_1 are fresh nonces.

It is assumed that an HCRA process has been performed before Service 1 contacts Service 2. In Figure 3 instance *A* is a session partner of *S*, and has registered with *SA*. When *A* tries to contact Service 2, it first sends a request (message (1)) to the factory service *F* of Service 2. *F* then generates a new instance *B* and sends the related information about *B* (message (2)) back to *A*. Next, *A* introduces *B* to *SA* (message (3)). After receiving the confirmation from *SA* (message (4)), *A* will start to communicate with *B* (messages (5) and (6)). During this process, the integrity of messages (1) and (2) needs to be protected by additional security channels (e.g. SSL, the secure conversation protocol, the secure message protocol etc.) as *B* is not yet a session partner during those steps. The integrity of messages (3), (4), (5), (6) is protected by shared secret keys distributed within *S*. For example, *A* can use its private key and the identifier of *B* to generate *K*(*A*,*B*) according to the Diffie-Hellman algorithm. *K*(*A*,*B*) is then used to generate the message authentication code of message (5). Similarly, *B* can use its private key and the identifier of *A* to generate *K*(*B*,*A*), which is identical to *K*(*A*,*B*). *K*(*B*,*A*) is then used to generate the MAC of message (6).

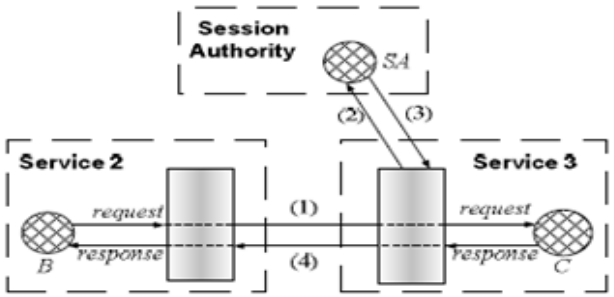


Figure 4: Protocol 2: Authenticating a session partner

Figure 4 illustrates Protocol 2: Authenticating a session partner. *B* and *C* are session partners of *S*, but *B* has not yet communicated with *C* before. First, *B* sends a request message (1) to *C*. *C* then sends message (2) to *SA* in order to check the identity of *B*. *SA* will send back a confirmation in message (3), confirming that *B* is a session partner of *S*. After receiving the confirmation, *B* will handle the request from *C* and send the result back. All the messages transferred during this process are encrypted by the secret key generated with the Diffie-Hellman algorithm. The details of the messages passed in Figure4 are presented as follows:

(1) $B \rightarrow C$: $Valid(Request, ID_B, ID_C, ID_S, N')_{K(B,C)}$
(2) $C \rightarrow SA$: $Valid(Query, ID_B, ID_C, ID_{SA}, ID_S, N'')_{K(C,SA)}$
(3) $SA \rightarrow C$: $Valid(SP(B,S), ID_{SA}, ID_C, ID_S, N''+1)_{K(SA,C)}$
(4) $C \rightarrow B$: $Valid(Response, ID_C, ID_B, ID_S, N'+1)_{K(C,B)}$
where N' and N'' are fresh nonces.

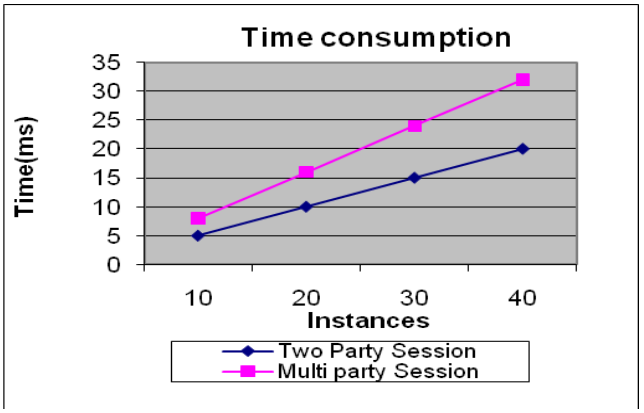
In Protocols 1 and 2, MACs are used to protect the integrity of the messages transported within a business session, and fresh nonces are used to guarantee that a message is not replayed.

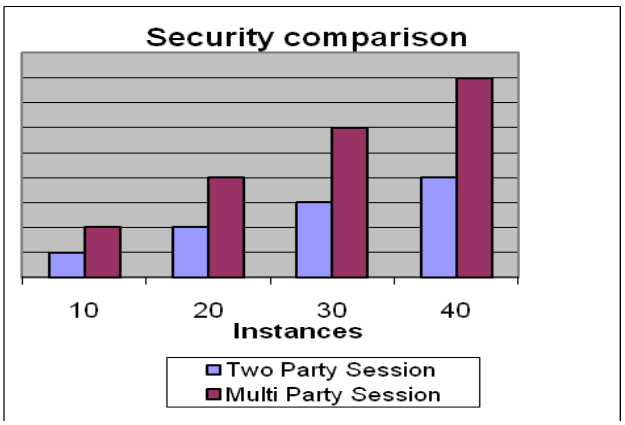
IMPLEMENTATION DETAILS

Beside the correctness analysis, we also need to examine whether our authentication system is feasible enough for practical real-world applications. Consequently, a series of experiments has been implemented to assess the overheads imposed by the authentication mechanisms and the scalability of our proposed system. Because the system is designed to be deployed on service-oriented middleware, we will evaluate the compatibility of our system with existing message-level security protocols.

We have implemented our idea using NetBeans 6.9.1 IDE. We have used JSP as our front end tool to create web pages and used Microsoft SQL server 2000 to maintain backend databases.

We made a comparison evaluation for security and performance overhead for authentication using two party and multi party session. The results of which are displayed in the following figures.





CONCLUSION

In practice, a dynamic business process may involve many applications and services which belong to different organizations and security realms. The dynamic authentication process between organizations could be highly complex and time-consuming if some intermediate authentication paths have to be created and credentials have

to be converted. When there is no existing authentication relationship in place between two organizations, it will be practically difficult for a system to enable any secure collaboration between services from the two organizations in a just-in-time fashion.

We have developed a new authentication system for multi-party service interactions that does not require credential conversion and the establishment of any authentication path between collaborative session partners. The system also offers the ability to identify individual service instances within a business session even if some instances in fact belong to the same service. Although the amount of communications between the partners of a session and the Session Authority is limited, the performance overhead imposed by it is indeed of some practical concern.

ACKNOWLEDGMENT

The authors wish to thank the anonymous reviewers for their valuable suggestions and comments which motivated us towards new scope in our research.

REFERENCES

[1]. S. Bajaj, G. Della-Libera, B. Dixon, M. Dusche, M. Hondo, M. Hur, C. Kaler, H. Lockhart, H. Maruyama, A. Nadalin, N. Nagaratnam, A. Nash, H. Prafullchandra, and J. Shewchuk, "Web Services Federation Language (WS-Federation)," available from http://msdn2. microsoft. com/ en-us/library/ ms951236.aspx, Jul.2003.

[2]. M. Burrows, M. Abadi, and R. Needham, "A Logic of Authentication," ACM Trans. on Computer Systems, Feb. 1990, pp. 18-36.

[3]. F. Cabrera, G. Copeland, T. Freund, J. Klein, D. Langworthy, D. Orchard, J. Shewchuk, and T. Storey, "Web Services Coordination (WS-Coordination)," available from http:// www. ibm. com/ developerworks/ library/ws-coor/, Aug. 2002.

[4]. I. Cervesato, A.D. Jaggard, A. Scedrov, and C. Walstad, "Specifying Kerberos 5 Cross-Realm Authentication," Proc. Workshop on Issues in the Theory of Security, Long Beach, California, USA, 2005, pp. 12 – 26.

[5]. N. Cook, S. Shirvastava, and S. Wheater, "Distributed Object Middleware to Support Dependable Information Sharing between Organisations," Proc. International Conference on Dependable Systems and Networks, Maryland, USA, Jun. 2002, pp, 249- 258.

[6]. K. Czajkowski, D. Ferguson, I. Foster, J. Frey, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke, W. Vambenepe, "The WS Resource Framework Version 1.0," available from http://www.globus.org/wsrf/specs/ws-wsrf.pdf, 3 May 2004.

[7]. D. Georgakopoulos and M. Hornick, "An Overview of Workflow Management: From Process Modelling to Workflow Automation Infrastructure," Distributed and Parallel Database, Springer, Mar. 2005, pp. 119-153.

[8]. Li Gong, "Increasing Availability and Security of an Authentication Service," IEEE J. Selected Areas in Communication, vol. 11, no. 5, June, 1993, pp. 657-662.

[9]. S. Hada and H. Maruyama, "Session Authentication Protocol for Web Services," Proc. Symposium on Application and the Internet, Jan. 2002, pp. 158-165.

[10]. M. Hondo, N. Nagaratnam, and A. J. Nadalin, "Securing Web Services," IBM Systems J., 2002.

[11]. M. Huhns and M. P. Singh, "Service-Oriented Computing: Key Concepts and Principles," IEEE Internet Computing, vol. 9, no. 1, Jan. 2005, pp. 75-81.

[12]. O. Kornievskaia, P. Honeyman, B. Doster, and K. Coffman, "Kerberized Credential Translation: A Solution to Web Access Control," Proc. 10th USENIX Security Symposium, Washington, DC, USA, Aug. 2001.

[13]. N. Li, W. Winsborough, and J.C. Mitchell, "Distributed Credential Chain Discovery in Trust Management," J. Computer Security, vol. 11, no. 1, 2003, pp. 35-86.

[14]. P. C. van Oorschot, "Extending Cryptographic Logics of Belief to Key Agreement Protocols," Proc. the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, Nov. 1993, pp. 233– 243.

[15]. S. Rafaeli and D. Hutchison, "A Survey of Key Management for Secure Group Communication," ACM Comput. Surveys, vol. 35, no. 3, Sep. 2003, pp. 309-329.

[16]. M. K. Reiter and S. G. Stubblebine, "Resilient Authentication Using Path Independence," IEEE Trans. Computers, vol. 47, no. 12, Dec. 1998, pp. 1351-1362.

[17]. W. Stallings, Cryptography and Network Security: Principles and Practices, Prentice Hall, Upper Saddle River, New Jersey, 1999.

[18]. M. Steiner, G. Tsudik, and M. Waidner, "Diffie- Hellman Key Distribution Extended to Group Communication," Proc. of the 3rd ACM Conference on Computer and Communications Security, New Delhi, India, Mar. 1996, pp. 31-37.

[19]. H. Sun, Y. Zhu, C. Hu, J. Huai, Y. Liu, and J. Li, "Early Experience of Remote and Hot Service Deployment with Trustworthiness in CROWN Grid," Proc. APPT, 2005, pp. 301-312.

[20]. C. K. Wong, M. G. Gouda, and S. S. Lam, "Secure Group Communications Using Key Graphs," Proc. ACM SIGCOMM '98 Conf. Applications, Technologies, Architectures, and Protocols for Computer Comm., 1998, pp. 68-79.

