Dynamic Authentication for Cross-Realm SOA-Based Business Processes

Jie Xu, *Member*, *IEEE Computer Society and IEEE*, Dacheng Zhang, Lu Liu, *Member*, *IEEE*, Xianxian Li

Abstract— Modern distributed applications are embedding an increasing degree of dynamism, from dynamic supply-chain management, enterprise federations, and virtual collaborations to dynamic resource acquisitions and service interactions across organizations. Such dynamism 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. The correctness of the protocol is formally analyzed and proven, and an empirical study is performed using two production quality Grid systems, Globus 4 and CROWN. The experimental results indicate that the proposed protocol and its implementation have a sound level of scalability and impose only a limited degree of performance overhead, which is for example comparable with those security-related overheads in Globus 4.

Index Terms— Authentication, inter-organizational security, multi-party interactions, Service-Oriented Architecture, Web services

1 Introduction

With the emergence of service-oriented technologies, dynamism and flexibility are becoming the core characteristics of modern inter-organizational business processes, such as business application integration, distributed auction services, and order processing [1, 2]. Within a service-oriented architecture (SOA), an organization may encapsulate and publish its applications as services, and select and interact at runtime with the services provided by other organizations. However, such dynamic interactions at runtime raise immediate problems of security, trust and dependability [3]. Until these problems are addressed and solved satisfactorily, the potential of automatic inter-organizational business processes will be severely restricted.

In a dynamic and distributed environment, it is often difficult for a complex business process to follow a static business specification. The execution order of its activities at runtime is usually unpredictable, and on some occasions, the actual execution of a process can be "one-of-a-kind" [4]. The applications and services involved in a complex business process are typically heterogeneous, provided by different organizations. Since each organization has its own security mechanisms and policies to protect its local resources, the business process has to operate amongst multiple, heterogeneous *security realms*. A security realm is a group of *principals* (e.g. people, computers, services) that are registered with an authentication authority and managed through a consistent set of security processes and policies for resource sharing. An authentication authority is a trusted principal that performs reliable authentication functions [5]. Authentication is a critical measure for any security realm. Before a principal is allowed to access the resources in a realm, its identity must be verified.

Existing cross-realm authentication mechanisms require either *federated authentication* by maintaining direct cross-realm authentication relationships between any pair of security realms (often costly or impractical) or additional *credential conversion* from one realm to another. In this paper we present a new solution for dynamically authenticating the services from different realms for SOA-based business processes at runtime. 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 on-the-fly trust relationships between the members of a business session, and (3) a set of protocols for multi-party session management, supported by formal analysis and em-

J. Xu is with School of Computing, University of Leeds, Leeds, West Yorkshire, LS2 9JT, UK. E-mail: j.xu@leeds.ac.uk
D. Zhang is with Beijing Research Centre of Huawei Technology, Shangdi, Beijing, 100085, China. E-mail: zhang_dacheng@hotmail.com
L. Liu is with School of Computing and Mathematics, University of Derby, Derby, Derbyshire, DE22 1GB, UK. E-mail: l.liu@derby.ac.uk
X. Li is with Faculty of Computer Science, Beihang University, Haidian, Beijing, China. E-mail:lixx@buaa.edu.cn

Manuscript received Oct 20, 2009. This work was supported in part by the UK EPSRC/BAE Systems NECTISE project (EP/D505461/1), the ESRC MoSeS project (RES-149-25-0034), the EPSRC WRG project (EP/F057644/1), the EPSRC CoLaB project (EP/D077249/1) and the Major Program of the National Natural Science Foundation of China (No.90818028).

pirical evaluation. The correctness of these protocols is formally verified using the well-known BAN logic for authentication [6, 7]. We have also designed and implemented an authentication system that employs the multi-party session protocols and allows service instances working inside a business session to authenticate each other based on a variety of credentials available, e.g. shared session keys and session memberships. This system has been incorporated into the CROWN-C Grid middleware [8]. Empirical evaluation has been performed to assess the system's scalability and runtime overheads.

The rest of paper is organized as follows. In Section 2 we discuss the fundamentals of constructing multi-party service interactions. Section 3 describes our proposed authentication protocols with formal proofs. In Section 4 we present an empirical evaluation using GT4 and CROWN middleware systems [9]. Section 5 discusses the related work while Section 6 concludes the paper.

2. Authentication across Heterogeneous Security Realms

Dynamic collaboration amongst different organisations is becoming increasingly common. Because principals could join or leave a collaborative business process in a highly dynamic and frequent fashion, it cannot be expected that a direct cross-realm authentication relationship always exists between each pair of the collaborating security realms. We consider two potential solutions in this section: federated authentication and credential conversion. These methods are then analyzed and compared with our proposed solution.

2.1 Federated Authentication

Federated authentication [10-13] offers a method for implementing a direct authentication relationship between different organizations and business partners. Examples include SAML [10], OpenID [11], Liberty [12], and Windows Cardspace [13]. However, maintaining a direct authentication relationship between any pair of the security realms involved is costly and often difficult to implement in practice. Federated authentication requires specific infrastructure support, and it also involves costly contract amendments and time-consuming activities for establishing mutual understandings between partners. Moreover, when a server needs to authenticate a chain of credentials submitted by a client, the server has to perform multiple, expensive digital signature verifications [14]. These are main reasons for the slow adoption of existing mechanisms for federated authentication.

2.2 Authentication-path of Credential Conversion

Another solution to this problem would be to locate some intermediate realms that serve as an authentication-path of credential conversion between the two separate realms that are to collaborate. However, the overhead of generating an authentication-path for two distributed realms is not trivial, which requires the cooperation from intermediate security realms. The process may involve a large number of extra operations for credential conversion as well as a long chain of invocations to intermediate services.

Moreover, such an authentication path of credential conversion between two security realms may not even exist. Fig. 1 illustrates an example of a collaborative business process where R_A and R_B are two security realms involved, and P_1, …, P_n are principals involved. An authentication path could be created from P_1 to P_n if R_B chooses to recommend the principals introduced from R_A to its federated security realms. However, R_A may not trust R_B enough, thereby deciding not to recommend any principal from R_B. When P_n attempts to contact P_1, P_n wouldn't be able to find a path for authentication. Without the existence of an authentication path, two realms to collaborate have to follow a more traditional and time-consuming method for building mutual trust, possibly with the support of contractual arrangements, multi-round negotiation, and human intervention.

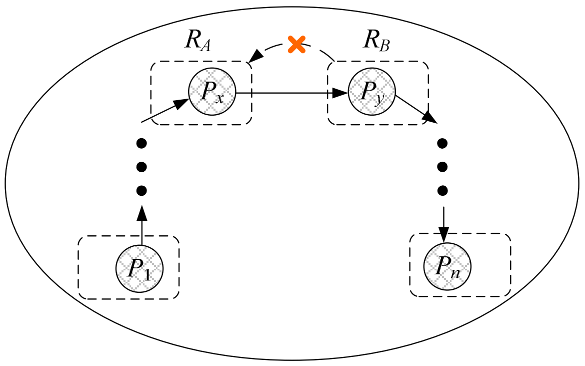

Fig. 1. A scenario of credential conversion between the two separate realms

We believe what is needed is a "dynamic" scheme for multi-party authentication. Unlike traditional "static" authentication protocols which assume pre-existing and fixed authentication relationships, a new scheme should be able to establish instantly at runtime an authentication relationship between any pair of principals that wish to collaborate. We will discuss and explain in the next section our solution for dynamic authentication based on the notion of multi-party business sessions.

3. Multi-Party Business Sessions

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 [15], 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 that session.

3.1 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. 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 [16].

The two-party session technique is both simple and 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 when the two-party session technique is applied directly to the construction of a multi-party session that has three or more session partners. Hada and Maruyama in [1] 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. This is because the two-party session technique does not have a mechanism for distinguishing a principal from a group of other principals.

In practice, the collaborating organizations involved in a business process execution often belong to different security realms. These realms may be heterogeneous, equipped with different security systems and mechanisms, and they are not necessarily interoperable. The two-party session technique does not address the issue with Heterogeneous Cross-Realm Authentication (HCRA), which typically requires credential conversion and the establishment of authentication paths.

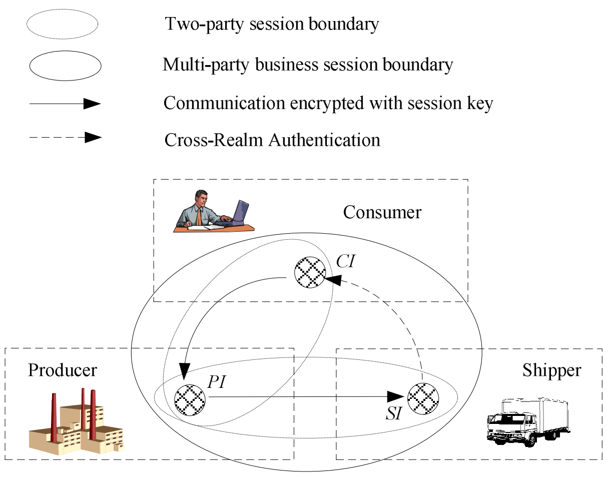
Two-party session boundary
Multi-party business session boundary
Communication encrypted with session key
Cross-Realm Authentication
Consumer
CI
Producer
PI
Shipper
SI

Fig. 2. A scenario of business process execution

Fig. 2 illustrates an example of a three-party business session but 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* might have to be performed by means of a new two party session if *SI* and *CI* do not know each other and belong to different security realms. This HCRA process is both costly and complex. In some cases it is practically impossible to perform HCRA in an automatic and on-the-fly manner, and human intervention will be needed. For a business session involved with n heterogeneous security realms, the HCRA process would have to be repeated $n \times (n-1)/2$ times to allow all possible partner interactions with the session.

3.2 Multi-Party Session

A multi-party session may have two or more session partners for an 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 normally 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 the session partners can make use of their shared session secrets and 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. 2 again. When *SI* attempts to contact *CI*, it does not need to authenticate itself with the local authentication system of *CI* because both *SI* and *CI* are already members of the same session. *SI* can simply use its session membership and/or shared session secrets to prove its identity to *CI*. This simplified authentication process is called Simplified Cross-Realm Authentication (SCRA). While the HCRA process has to be repeated up to $(n-1)$ times (for the introduction of any new partners) for a multi-party session with n security realms, up to $(n-1) \times (n-2)/2$ authentication processes between any existing partners can be simplified as SCRA, thereby reducing both cost and complexity significantly.

There are different ways to start a multi-party business session, e.g. one or a group of services [18] may serve as a starter to initiate a session. Any existing session partner of the session may introduce new members into the session. If HCRA between the existing partner and the potential new member has been performed based on the established trust relationship between them, the authentication process for the session will be further simplified. In addition, there is some recent progress in developing protocols and mechanisms for HCRA based on automatic negotiation at runtime [24]. The work could be combined with SCRA to reduce further the cost and complexity of dynamic cross-realm authentication.

It is important to notice that multi-party business sessions are also a useful design abstraction for controlling and coordinating multi-party interactions and collaborations. While it is convenient to use this abstraction to develop our proposed authentication system for automatic SOA-based business processes, our system could be combined with other models for multi-party interactions such as atomic actions [3] and process groups [16]. However, regardless of the model used, a multi-party authentication 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 [1].

3.3 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 a 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.

3.4 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 [1] 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 with forward security techniques [19] 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. [20, 21]).

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 [22]). This scheme is more costly but it avoids the issue with key revocation.

3.5 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 [23], 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 [24] 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.

3.6 Network Threats

There are major security threats to SOA-based business processes that need to be addressed, including identity theft and replay attacks. Identity theft refers to a fraud that involves someone pretending to be someone else for an illegal purpose. This type of security threat is in fact addressed by our proposed approach. In our authentication protocol, a session partner is required to generate its security key pair locally, and keep the private key in a secure place. Without having access to the correct private key, it is computationally difficult for an attacker to generate the required symmetric key using the Diffie-Hellman algorithm in order to impersonate an honest session partner.

A replay attack is an attack where an authentication session is replayed by an attacker for granting access. In order to address this type of threat, our protocol assigns a token to every message transported amongst session partners and session authorities. The token consists of a constantly increasing sequence number, the session ID, and the identifiers of the sender and the recipient. If a message is resent to the original recipient, the attack will be detected by the recipient based on the sequence number. If the message is resent to other partners in the session, the message will be rejected by the partners as the recipient information of the message is incorrect. Moreover, if the message is sent to a different session, the attack will be detected easily as the session ID is different and incorrect.

4. Authentication Protocols

In this section we will 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. The formal analysis enables us 1) to re-examine assumptions used to develop our authentication protocols, 2) to verify whether the objectives of our protocols are achieved through the intended actions, and 3) to analyze whether the cryptographic methods used are able to prevent impersonation and replay attacks.

4.1 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 bound with the "real-world" identity (e.g. company name) 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 collaborating partners of a session.

Fig. 3 illustrates how the authentication system performs multi-party session authentication and management using the example of Fig. 2. 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* and *SI* can use the Diffie-Hellman algorithm to agree upon a shared secret key for further communications.

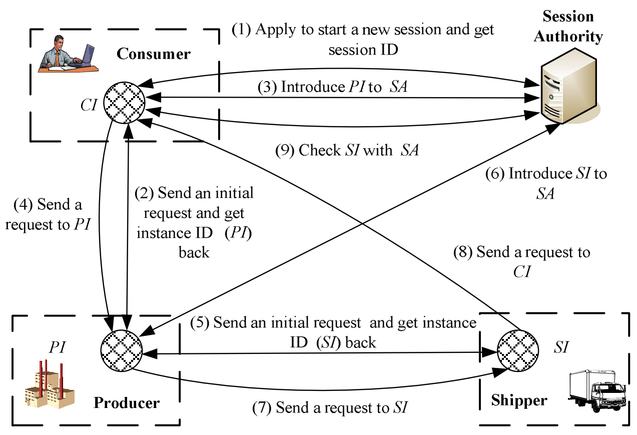

Fig. 3. A business scenario

4.2 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) [6, 7]. Protocol 1 is concerned with the introduction of a new session partner, and Protocol 2 performs authentication between two existing session partners. Notice that all messages of the protocols are enveloped in the XML documents and transported by the WS-protocols (e.g. SOAP). It implies that types of the messages are tagged. We therefore denote abstractly the messages as different types of symbols [25]. 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 [6]).

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. ($a^{Pri(A)}$ mod p) = ID_A
X, Y	range over statements
(M, N)	composite message composed of messages M and N
$K_{(A, B)}$	secret key generated with $Pri(A)$ and $Pub(B)$; $K_{(A, B)} = (Pub(B))^{Pri(A)} = a^{Pri(A)\ Pri(B)}$ mod p; $K_{(A, B)} = K_{(B, A)}$
$MAC(M)_K$	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* [6], i.e. its corresponding $Pri(A)$ will never be discovered by any oth-

	er principals and *Pub*(*A*) is not weak (e.g. *Pub*(*A*)=1)
$\#M$	*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
$A \xleftrightarrow{K^{-}_{(A,B)}} B$	$K_{(A,B)}$ is *A*'s secret key to be shared with *B*. No third principal aside from *A* and *B* can deduce $K_{(A,B)}$, but *A* has not yet get confirmation from *B* that *B* knows $K_{(A,B)}$
$A \xleftrightarrow{K^{+}_{(A,B)}} B$	$K_{(A,B)}$ is a key held by *A*. No third principal aside from *A* and *B* can deduce $K_{(A,B)}$. And *A* has received key confirmation from *B* which indicates that *B* actually knows $K_{(A,B)}$
$A \mid\equiv X$	*A* believes that statement *X* is true
$A \mid\Rightarrow X$	*A* is an authority on *X*, i.e. *A* has jurisdiction over *X*
$A \triangleleft M$	*A* receives message *M* from somebody

Fig. 4 illustrates Protocol 1: Accepting a new session partner. Our protocol conforms to the WSResource Framework (WSRF) specification [26], where a service is associated with a factory service *F* that generates service instances.

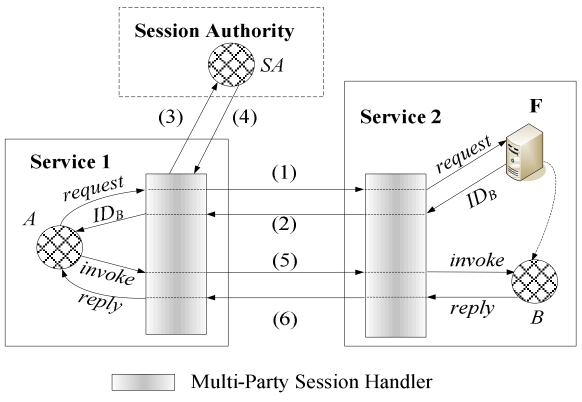

Fig. 4. Protocol 1: Accepting a new session partner

The details of the messages transported within Fig. 4 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 Fig. 4 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).

Fig. 5 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 Fig. 5 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.

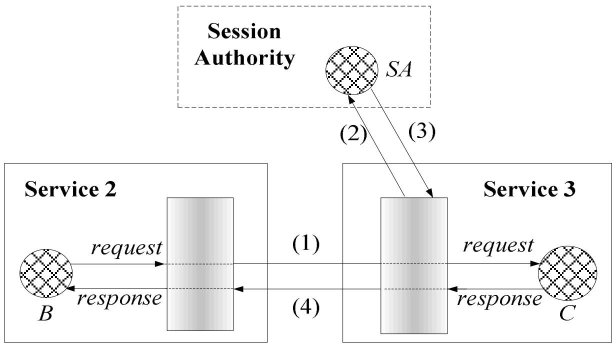

Fig. 5. Protocol 2: Authenticating a session partner

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.

4.3 Correctness Proofs

In this section we use the extended BAN logic [6, 7] to analyze formally the correctness of Protocols 1 and 2. We first introduce some deduction rules to be used by our correctness proofs. These rules are specified in [6, 7].

Rules:

Rule 1: $A \mid\equiv (X, Y) \Rightarrow A \mid\equiv X$ and $A \mid\equiv Y$
A believes a set of statements if and only if A believes every individual statement respectively.

Rule 2: $A \mid\equiv \# M \Rightarrow A \mid\equiv \#(M, N)$ and $A \mid\equiv \#(N, M)$
The whole message is believed to be fresh if a part of a message is believed to be fresh.

Rule 3: $A \mid\equiv B \mid\Rightarrow X, A \mid\equiv B \mid\equiv X \Rightarrow A \mid\equiv X$
This inference rule states that if A believes that B has a control over statement X, and if A believes that B believes X, then A should believe X.

Rule 4: $A \mid\equiv \uparrow Pub(A), A \mid\equiv \uparrow Pub(B) \Rightarrow A \mid\equiv A \xleftrightarrow{K^{-}_{(A,B)}} B$
This rule is derived from R31 in [6]. In this rule, if A has B's public key and believes that the public keys of A and B are both good, A can believe that the secret is shared with no party other than B although unconfirmed.

Rule 5: $A \mid\equiv A \xleftrightarrow{K^{-}_{(A,B)}} B$, $A \lhd Valid(X)_{K(A,B)}$, $A \mid\equiv \# X \Rightarrow A \mid\equiv A \xleftrightarrow{K^{+}_{(A,B)}} B$ This rule derived from R32 in [6].

Rule 6: $A \lhd Valid(X)_{K(A,B)}$, $A \mid\equiv A \xleftrightarrow{K^{+}_{(A,B)}} B$, and $A \mid\equiv \# X \Rightarrow A \mid\equiv B \mid\equiv X$

Lemma 1 $A \lhd Valid(M)_{K(A,B)}$, $A \mid\equiv A \xleftrightarrow{K^{+}_{(A,B)}} B$, and $A \mid\equiv \# M$, then $A \mid\equiv B \mid\equiv M$.

Proof: This lemma can be deduced directly from Rule 6.

4.3.1 Protocol 1: Accepting a New Session Partner

Security goals of the protocol of accepting a new session partner include 1) accepting B as a new partner and 2) building a confirmed secret key to be shared between A and B. As illustrated in Fig. 4, this protocol conforms to the WS-Resource Framework (WSRF) specification [26], where a service is associated with a factory service F that generates service instances. The security goals are formally described as follows:

$SA \mid\equiv SP(B,S)$, $A \mid\equiv A \xleftrightarrow{K^{+}_{(A,B)}} B$, and $B \mid\equiv B \xleftrightarrow{K^{+}_{(B,A)}} A$

Additionally, the assumptions of this protocol are formally described as follows:

$SA \mid\equiv \uparrow Pub(SA)$, $A \mid\equiv \uparrow Pub(A)$, $B \mid\equiv \uparrow Pub(B)$, $SA \mid\equiv \uparrow Pub(A)$, $A \mid\equiv \uparrow Pub(SA)$, $A \mid\equiv \uparrow Pub(B)$, $B \mid\equiv \uparrow Pub(A)$, $SA \mid\equiv \# N$, $A \mid\equiv \# N_1$, $B \mid\equiv \# N_1$, $SA \mid\equiv A \mid\Rightarrow SP(B, S)$

It is the responsibility of SA to decide whether to accept an instance (e.g. B) as a session partner following certain policies. This assumption is in fact based on the simplest policy, that is, SA will accept any instance recommended by an existing session partner (e.g. A) as a new session partner. Other policies may include, for example, that only a particular set of session partners have privileges to introduce new session partners to SA; a new session partner can be accepted only when a majority of session partners vote to accept it.

To prove the correctness of the protocol, it is necessary to show whether its security goals can be satisfied after running the protocol under the stated assumptions. Therefore the following theorem is proposed.

Theorem 1 The goals of the protocol of accepting a new session are satisfied under the assumptions of the protocol.

Proof: It is needed to deduce $SA \mid\equiv SP(B,S)$, $A \mid\equiv A \xleftrightarrow{K^{+}_{(A,B)}} B$, and $B \mid\equiv B \xleftrightarrow{K^{+}_{(B,A)}} A$ from the assumptions of the protocol.

The third step of this protocol implies that SA $\lhd$ $Valid(SP(B,S), ID_B, ID_A, ID_{SA}, ID_S, N)_{K(A, SA)}$. We obtain $SA \mid\equiv \#(SP(B,S), ID_B, ID_A, ID_{SA}, ID_S, N)$ by the assumption $SA \mid\equiv \# N$ and Rule 2. From $SA \mid\equiv \uparrow Pub(SA)$ and $SA \mid\equiv \uparrow Pub(A)$, it follows that $SA \mid\equiv A \xleftrightarrow{K^{-}_{(A,SA)}} SA$ by Rule 4. Then, from $SA \mid\equiv A \xleftrightarrow{K^{-}_{(A,SA)}} SA$, $SA \lhd Valid(SP(B,S), ID_B, ID_A, ID_{SA}, ID_S, N)_{K(A, SA)}$, and $SA \mid\equiv \#(SP(B,S), ID_B, ID_A, ID_{SA}, ID_S, N)$, it yields that $SA \mid\equiv A \xleftrightarrow{K^{+}_{(A,SA)}} SA$ by Rule 5. Furthermore, we can deduce that $SA \mid\equiv A \mid\equiv (SP(B,S), ID_B, ID_A, ID_{SA}, ID_S, N)$ and $SA \mid\equiv A \mid\equiv SP(B,S)$ by Lemma 1 and Rule 1. Therefore, from the assumption $SA \mid\equiv A \mid\Rightarrow SP(B, S)$ and Rule 3, it follows that $SA \mid\equiv SP(B,S)$.

From $B \mid\equiv \uparrow Pub(B)$ and $B \mid\equiv \uparrow Pub(A)$, we have $B \mid\equiv B \xleftrightarrow{K^{-}_{(B,A)}} A$ by Rule 4. Besides, from the sixth step of the protocol and the assumption $B \mid\equiv \# N_1$, it follows that $B \lhd Valid(Invoke, ID_A, ID_B, ID_S, N_1)_{K(A, B)}$ and $B \mid\equiv \#(Invoke, ID_A, ID_B, ID_S, N_1)$. Consequently, we obtain $B \mid\equiv B \xleftrightarrow{K^{+}_{(B,A)}} A$ by Rule 5.

From $A \mid\equiv \uparrow Pub(A)$ and $A \mid\equiv \uparrow Pub(B)$, we have $A \mid\equiv A \xleftrightarrow{K^{-}_{(A,B)}} B$ by Rule 4. Besides, from the sixth step of the protocol and the assumption $A \mid\equiv \# N_1$, it follows that $A \lhd Valid(Reply, ID_B, ID_A, ID_S, N_1{+}1)_{K(B, A)}$ and $A \mid\equiv \#(Reply, ID_B, ID_A, ID_S, N_1{+}1)$. Consequently, we obtain $A \mid\equiv A \xleftrightarrow{K^{+}_{(A,B)}} B$ by Lemma 2. $B \mid\equiv B \xleftrightarrow{K^{+}_{(B,A)}} A$ can also be deduced through a similar procedure. Hence the theorem.

4.3.2 Protocol 2: Authenticating a Session Partner

The security goals of the protocol of authenticating a session partner are 1) verifying whether a principal is a session partner, and 2) building a confirmed secret key to be shared between the session partners. Formal expression of the security goals are presented as follows:

$C \mid\equiv SP(B, S)$, $B \mid\equiv B \xleftrightarrow{K^{+}_{(B,C)}} C$, and $C \mid\equiv C \xleftrightarrow{K^{+}_{(B,C)}} B$.

The formal descriptions of the assumptions are:

$C \mid\equiv \uparrow Pub(C)$, $SA \mid\equiv \uparrow Pub(SA)$, $B \mid\equiv \uparrow Pub(B)$, $C \mid\equiv \uparrow Pub(SA)$, $B \mid\equiv \uparrow Pub(C)$, $SA \mid\equiv \uparrow Pub(C)$, $C \mid\equiv \uparrow Pub(B)$, $B \mid\equiv \# N'$, $C \mid\equiv \# N'$, $C \mid\equiv \# N''$, and $C \mid\equiv SA \mid\Rightarrow SP(B, S)$

The correctness of this protocol is stated in the following theorem.

Theorem 2 The goals of the protocol of authenticating a session partner are satisfied under the above assumptions.

Proof: By Rule 4, $C \mid\equiv \uparrow Pub(C)$ and $C \mid\equiv \uparrow Pub(SA)$ imply that $C \mid\equiv C \xleftrightarrow{K^{-}_{(C,SA)}} SA$. We obtain $C \lhd Valid(SP(B, S), ID_{SA}, ID_C, ID_S, N''{+}1)_{K(SA,C)}$ in the third step of the protocol, and $C|\equiv\#(SP(B, S), ID_{SA}, ID_C, ID_S, N''+1)$ by the assumption $C|\equiv\#N''$ and Rule 2. It follows that $C|\equiv C \xleftrightarrow{K^{+}_{(C,SA)}} SA$ by Lemma 2. By Lemma 1 we obtain that $C|\equiv SA \models (SP(B, S), ID_{SA}, ID_C, ID_S, N''+1)$, and thus $C|\equiv SA \models SP(B, S)$ by Rule 1. Since $C|\equiv SA \mid\Longrightarrow SP(B, S)$, we have $C|\equiv SP(B, S)$ by Rule 3.

From the assumption $C|\equiv\uparrow Pub(C)$ and $C|\equiv\uparrow Pub(B)$, it follows that $C|\equiv C \xleftrightarrow{K^{-}_{(C,B)}} B$ by Rule 4. Since $C \lhd Valid(Request, ID_B, ID_C, ID_S, N')_{K(B,C)}$ and $C|\equiv\#(Request, ID_B, ID_C, ID_S, N')$ which is derived from $C|\equiv\#N'$ by Rule 2, then $C|\equiv C \xleftrightarrow{K^{+}_{(C,B)}} B$ by Rule 5. $B|\equiv B \xleftrightarrow{K^{-}_{(B,C)}} C$ can be deduced from $B|\equiv\uparrow Pub(C)$ by a similar approach. Hence the theorem.

4.3.3 Two Trivial Cases

For the completeness of our description and discussion, we outline in this section two simple protocols for a partner to leave from a session and for ending a session. As a two-party session has only two partners, a two-party session will end automatically when a partner leaves from the session. A business session with more than two partners may continue to operate after a partner has left, as long as the keys used by that partner are revoked properly.

In practice, an SA can generate a public key pair and distribute the public key to all the existing session partners. Whenever a partner is to leave from the session, the SA can generate a `SessionPartnerLeaving` message and sign the message with its private key. The message and the signature are then sent to all the session partners that used to communicate with the leaving partner. All the shared keys between them can be therefore reinvoked. This solution is particularly efficient when the leaving partner has a large number of contacts within the session.

A multi-party session could terminate in two circumstances. The session ends when all the partners have left. It is also possible that there are still session partners operating in the session but the session has to end for some reasons, e.g. in the occurrence of exceptions. The exception should be signalled to the SA. The SA will then send a `SessionEnding` message to all the partners still in action and dispose the associated SA instance.

5. Empirical Evaluation

Beside analytic assessments, the feasibility of the proposed authentication system in real-world applications also needs to be examined. Consequently, a series of experiments has been conducted using two production-quality Grid middleware systems in order to assess:

1) *scalability* of our multi-party authentication system,
2) *compatibility* of the system with other common message-level security protocols, and
3) *runtime overheads* of the mechanism under different conditions.

Two experimental systems have been developed. The first experimental system (*ES*1 for short) consists of an *SA* service and three experimental services. As illustrated in Fig. 6, a client in *ES*1 first initiates a business session, and then three experimental services repeatedly invoke each other until a particular amount of service instances have been generated and accepted into the business session. The second experimental system (*ES*2 for short) only consists of three experimental services. In the experiments, the experimental services of *ES*2 invoke each other repeatedly until a particular amount of service instances has been generated.

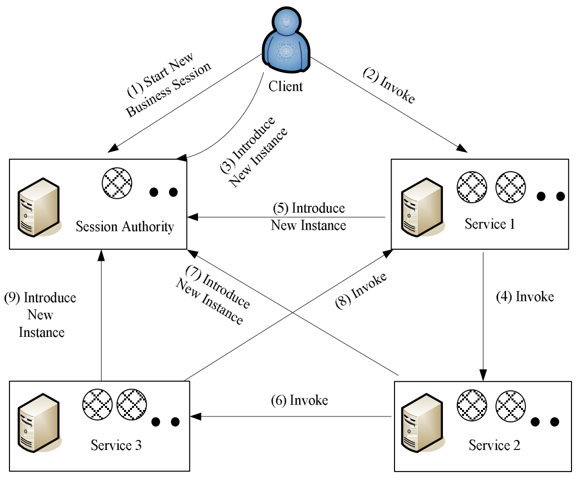
Client
(1) Start New Business Session
(2) Invoke
(3) Introduce New Instance
Session Authority
Service 1
(5) Introduce New Instance
(4) Invoke
(7) Introduce New Instance
(8) Invoke
(9) Introduce New Instance
(6) Invoke
Service 3
Service 2

Fig. 6. Experimental system with *SA*

*ES*1 is used to simulate distributed applications that use the multi-party authentication system whilst *ES*2 simulates distributed applications without the multiparty authentication system. The experimental results of *ES*2 are used as benchmarks. By comparing the experimental results obtained from both experimental systems, we can assess the overheads imposed by our authentication system.

Our experimental systems are implemented on a Grid service middleware system in which a Web service is associated with a factory service which is in charge of the generation and the management of resources. In the Grid, Web services are stateless, and state information is stored in resources. A service instance can be thus located when the corresponding resource is found. In *ES*1, the identifier of an instance and its associated private key are stored within a resource, and the instance identifier is identical to the resource identifier.

5.1 Worst Case Assessment

We have deployed the experimental systems both on a single computer and on a distributed system with multiple computers. Particularly, we deploy the experimental systems on a single computer for two reasons.

- In order to evaluate precisely the overhead introduced by the authentication protocols (e.g. generating key pairs, generating MACs for messages etc.), we need to remove the influence introduced by the time consumption of transporting messages.
- Deploying the experimental systems on a single computer can help us to evaluate the performance of the systems in the strictest environment where all the opera-

tions are executed sequentially. In this sub-section all the experiments are deployed on a single computer unless stated otherwise.

In this experiment, more than 24,000 instances are generated and introduced to the session authority. The experiment results indicate that the time consumed in accepting new service instances into a session is proportional to the number of the newly accepted instances shown in Fig. 7, until over 16,000 instances are accepted. The performance decreases afterwards, and the system finally stops due to a lack of memory space.

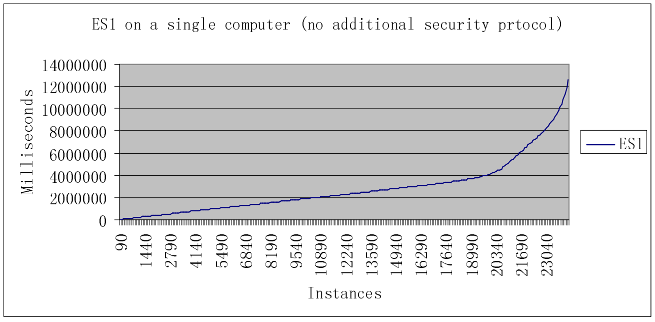

Fig. 7. *ES*1 deployed on GT4

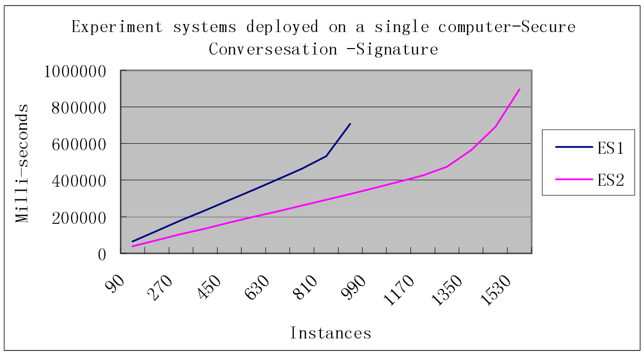

Fig. 8. *ES*1 and *ES*2 integrated with Secure Conversation on GT4

In addition, *ES*1 and *ES*2 have been combined with two message-level security protocols, *Secure Conversation* and *Secure Message*, provided by GT4. In the experiment, the secure conversation protocol is used to generate signatures for the messages transported in *ES*1. As illustrated in Fig. 8, the time consumption of the experimental system starts to increase non-linearly after over 720 instances are accepted. The same phenomenon occurs in *ES*2 after over 1040 instances are accepted when *ES*2 is integrated with the secure conversation protocol.

As illustrated in Figs 8 and 9, the time consumption of *ES*1 is about twice that of *ES*2 although *ES*1 provides sufficient support for session authentication operations. These experimental results indicate that the scalability of the authentication mechanism varies, when integrated with different message-level security protocols.

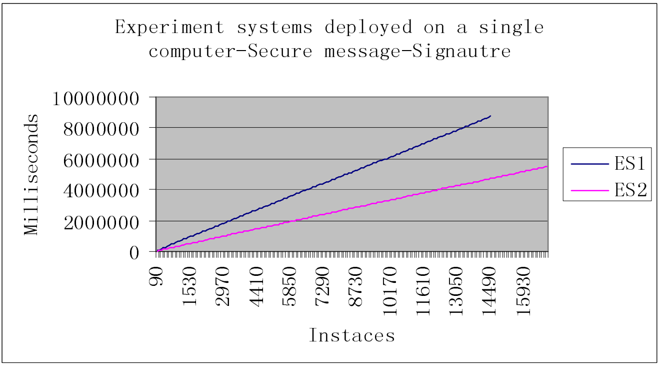

Fig. 9. *ES*1 and *ES*2 integrated with Secure Message on GT4

Fig. 10 shows the overhead imposed by different security mechanisms. From the results, we can see that the proposed multi-party authentication mechanism spends more time on key generation and distribution than the single key mechanism. However, the overhead introduced by the authentication mechanism is still comparable to the standard security mechanisms used in GT4.

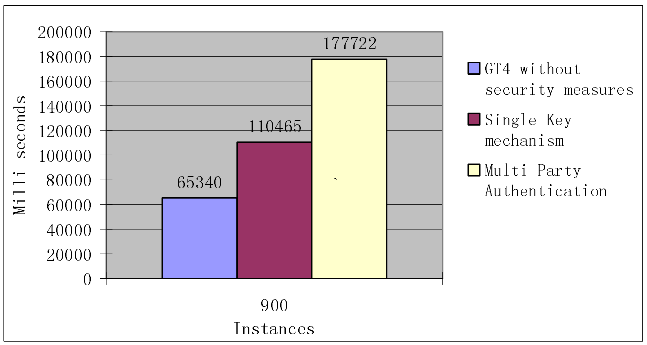

Fig. 10. Overhead imposed by security mechanisms

In order to avoid generating GT4-oriented results only, this design is also implemented in the large-scale CROWN (China Research and Development environment Over Wide-area Network) Grid [9]. CROWN is a practical Grid system, aiming to promote the utilisation of valuable resources and cooperation of researchers nationwide and world-wide. As illustrated in Fig. 11, the experimental results show that time consumption of *ES*1 is about twice that of *ES*2. This is very similar to what have been observed in the GT4-based experiments. However, as the computer used by CROWN is much more powerful, the performance of the experimental systems is much better. The experimental systems can stay in a stable state until over 260,000 instances are accepted.

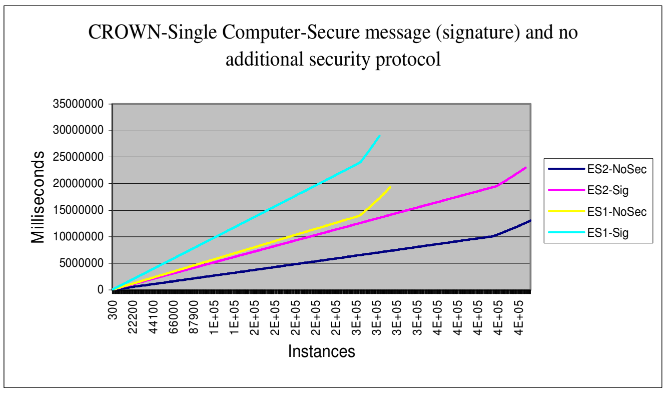

Fig. 11. *ES*1 and *ES*2 on CROWN

5.2 Distributed Deployment

In order to evaluate the scalability of the system in a more realistic and distributed environment, the experimental systems were deployed on a distributed computer system. The performance of the experimental systems is improved significantly when deployed in a distributed environment with multiple computers. In the GT4-based experiments (Fig. 12), the time consumption of *ES*1 without combining other security protocols increases linearly until more than 70,000 new instances are accepted.

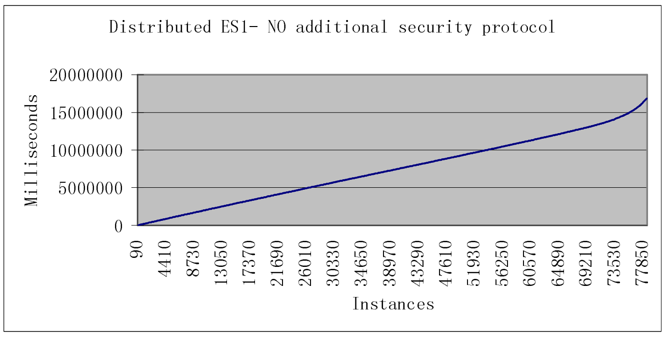

Fig. 12. *ES*1 deployed on GT4

Fig. 13 shows the proportions of time consumed by different operations in the experiment. In generating 900 instances, about 38% of the time is spent on key generation, about 25% of the time is used by the additional operations introduced by the multi-party authentication protocol, and about 37% of the time is spent on the essential operations of service invocations, e.g. instance generation and message transfer.

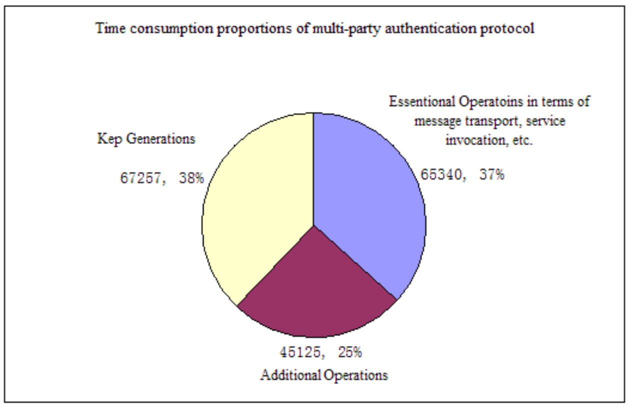

Fig. 13. Time consumption proportions

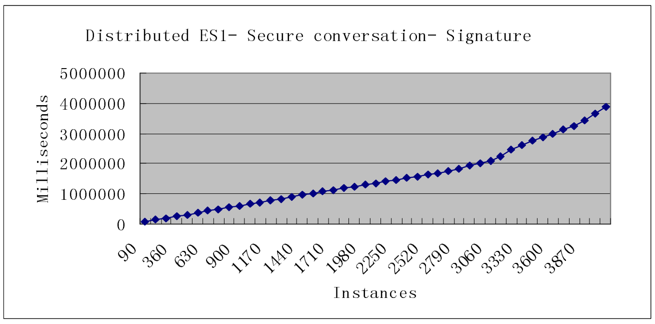

Fig. 14. *ES*1 integrated with secure conversation protocol on GT4

When combined with the secure conversation protocol (Fig. 14), *ES*1 executes stably until over 3,000 instances are accepted. As an example of the distributed deployment, the experimental results based on CROWN are presented in Fig. 15. The time consumption of *ES*1 is proportional to the number of newly accepted instances until over 300,000 instances are accepted.

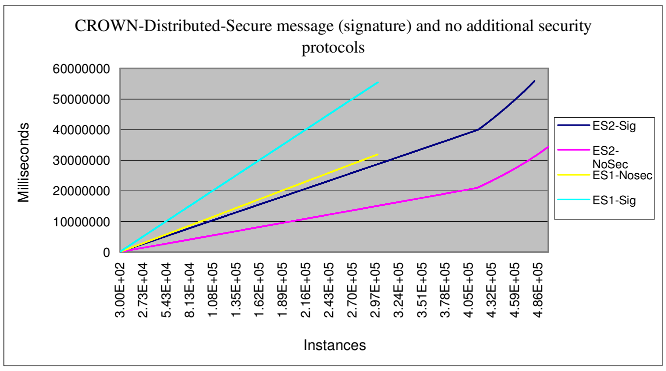

Fig. 15. Distributed *ES*1 and *ES*2 on CROWN

According to the experimental results, we discovered that memory space is a critical factor that affects the performance of the experimental systems. Information about session partners needs to be recorded by the *SA*, and the resources of experimental services also need to be stored in memory. When an experimental system is deployed in a distributed environment, the *SA* and the experimental services do not need to utilize the same limited memory. The performance of experimental systems therefore becomes much better. As presented in Fig. 16, the performance of the experimental system suddenly worsens near the end of the experiment deployed on GT4. This is because the operating system detects a lack of physical memory and attempts to move data from physical memory to swap space.

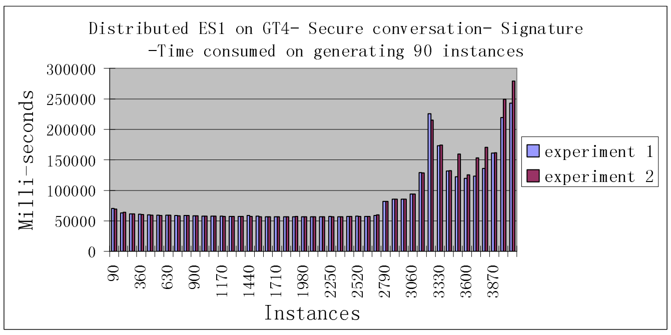

Fig. 16. Performance of *ES*1 on GT4

Another factor is the size of log files. During experiments, the experimental systems record state information in log files. When the size of log file increases, experimental systems need to take a large amount of memory and time on reading and writing the log files. Due to the behavior of the Java garbage collection mechanism, the consumed memory is sometimes not released in a timely manner, and the performance of the experimental system is thus affected.

6. Related Work

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 [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 heterogeneous authentication mechanisms, credential conversion mechanisms are widely used in many existing solutions. The work in [27] presents two types of credential translator services, KCA which translates Kerberos credentials to PK credentials, and KCT which translates PK credentials to Kerberos credentials.

Another example is PGP/X.509 [28]. In PGP, by identifying a chain of intermediaries, the receiver of a message can authenticate the sender even if they did not know each other. In [29], Jokl et al. discuss the authentication issues crossing different Grid security realms where independent CAs are applied. The solution proposed in the paper adopts bridge CAs to connect the CAs in different security realms so that a certificate can be validated through a path which may cross multiple security realms. Reiter and Stubblebine in [30] 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 [30]. 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 [31, 32]. 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. 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 have proposed a specification: WS-Coordination [24]. WS-Coordination describes an extensible framework for supporting the coordination of the actions in distributed applications. However, WS-Coordination 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.

7. Conclusions

In a Web Service context, a dynamic business process may involve many applications and services from different organisations and security realms, which are combined at runtime and collaborate in a peer-to-peer way. The dynamic authentication process between organisations could be highly complex and time-consuming since intermediate authentication paths need to be created at runtime to dynamically covert credentials from different security realms. If there is no existing authentication relationship in place between two organisations, it will be practically difficult for a system to enable any secure collaboration between services from the two organisations in a just-in-time fashion.

In response to this challenge, we have developed a new authentication system for multi-party service interactions that does not require credential conversion and establishment of authentication paths 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. We have therefore conducted a set of comprehensive experiments to assess the overhead on two service-oriented Grid systems. The experimental results were collected in a realistic and distributed setting which can accommodate concurrently more than 300K service instances. The main results show that the overhead imposed by our authentication system is comparable with the overheads caused by the standard security mechanisms used in those Grid middleware systems. An interesting future question is how heterogeneous security realms agree upon the usage of secret keys within a session [33]. We are developing a negotiation protocol to address this issue [34].

ACKNOWLEDGMENTS

The work reported in this paper has been supported in part by the UK EPSRC/BAE Systems NECTISE project (EP/D505461/1), the ESRC MoSeS project (RES-149-25-0034), the EPSRC WRG project (EP/F057644/1), the EPSRC CoLaB project (EP/D077249/1), the Major Program of the National Natural Science Foundation of China (No.90818028), and China 973 Fundamental R&D Program (No. 2011CB302600).

REFERENCES

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

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

[3] J. Xu, B. Randell, A. Romanovsky, R. J. Stroud, A. F. Zorzo, E. Canver, and F. v. Henke, "Rigorous Development of a Fault-tolerant Embedded System Based on Coordinated Atomic Actions," *IEEE Transactions on Computers, Special Issue on Fault-Tolerant Embedded Systems*, vol. 55, 2002, pp. 164-179,.

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

[5] J. D. Clercq, "Single Sign-On Architectures," in *International Conference, InfraSec 2002*, Bristol, UK, 2002, pp. 40-58.

[6] P. V. Oorschot, "Extending Cryptographic Logics of Belief to Key Agreement Protocols," in *the 1st ACM Conference on Computer and Communications Security*, Fairfax, Virginia, USA, 1993, pp. 233–243.

[7] M. Burrows, M. Abadi, and R. Needham, "A Logic of Authentication," *ACM Trans. on Computer Systems,* vol. 8, Feb. 1990, pp. 18-36.

[8] P. Townend, J. Huai, J. Xu, N. Looker, D. Zhang, J. Li, and L. Zhong, "CROWN-C: A High-Assurance Service-Oriented Grid Middleware System," *IEEE Computer,* vol. 41, 2008, pp. 33-38.

[9] 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," in *6th International Workshop on Advanced Parallel Processing Technologies (APPT 2005),* Hong Kong, China, 2005.

[10] P. Mishra, "Differences between OASIS Security Assertion Markup Language (SAML) V1.1 and V1.0. OASIS Draft". Available from:
http://www.oasis-open.org/committees/download.php/3412/sstc-saml-diff-1.1-draft-01.pdf,

[11] D. Fox, "Personal Theories of Teaching," *Studies in Higher Education,* vol. 8, 1983, pp. 151-163.

[12] W3C, "Web Services Choreography Description Language Version 1.0". Available from: http://www.w3.org/TR/ws-cdl-10/#Introduction,

[13] V. Bertocci, G. Serack, and C. Baker, *Understanding Windows CardSpace: An Introduction to the Concepts and Challenges of Digital Identities*: Addison-Wesley, 2007.

[14] J. Li, N. Li, X. Wang, and T. Yu, "Denial of Service Attacks and Defenses in Decentralized Trust Management," *International Journal of Information Security,* vol. 8, 2009, pp. 89-101.

[15] M. P. Papazoglou and D. Georgakopoulos, "Service-Oriented Computing," *Communications of the ACM,* vol. 46, Oct. 2003.

[16] L. Gong, "Increasing Availability and Security of an Authentication Service," *IEEE Journal on Selected Areas in Communication,* vol. 11, June 1993, pp. 657-662.

[17] W. Stallings, *Cryptography and Network Security (2nd ed.): Principles and Practice*: Prentice-Hall, Inc., 1999.

[18] J. Huai, Y. Zhang, X. Li, and Y. Liu, "Distributed Access Control in CROWN Groups," in *The 34th International Conference on Parallel Processing (ICPP 2005),* Norway, 2005.

[19] G. Itkis, "Forward Security: Adaptive Cryptography – Time Evolution," in *Handbook of Information Security*: John Wiley and Sons, 2006.

[20] S. Rafaeli and D. Hutchison, "A Survey of Key Management for Secure Group Communication," *ACM Computing Surveys,* vol. 35, Sep.2003, pp. 309-329.

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

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

[23] K. P. Birman, T. A. Joseph, T. Raeuchle, A. E. Abbadi, and A. E. ABBADI, "Implementing Fault-Tolerant Distributed Objects," IEEE Transactions on Software Engineering, vol. SE-11, 1985.

[24] 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/,

[25] J. Heather, G. Lowe, and S. Schneider, "How to prevent type flaw attacks on security protocols," in 13th IEEE Computer Security Foundations Workshop, 2000, pp. 255-268.

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

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

[28] P. R. Zimmermann, The Official PGP User's Guide. MA, USA: MIT Press Cambridge, 1995.

[29] J. Jokl, J. Basney, and M. Humphrey, "Experiences Using Bridge CAS for Grids," in UK Workshop on Grid Security Experiences, 2004.

[30] M. K. Reiter and S. G. Stubblebine, "Resilient Authentication Using Path Independence," IEEE Trans. Computers, vol. 47, December 1998, pp. 1351-1362.

[31] 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,

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

[33] D. Zhang, "Dynamic authentication for multi-party service interactions," in PhD Thesis, School of Computing: University of Leeds, 2008.

[34] J. Li, D. Zhang, J. Huai, and J. Xu, "Context-aware Trust Negotiation in Peer-to-Peer Service Collaborations," Peer-to-Peer Networking and Applications, vol. 2, 2009, pp. 164 - 177.

Prof Jie Xu is Chair of Computing at the University of Leeds (UK), Director of the Institute for Computational and Systems Science at Leeds, and Director of the EPSRC WRG e-Science Centre involving the three White Rose Universities of Leeds, York and Sheffield. He has worked in the field of Distributed Computer Systems for over twenty-five years and had industrial experience in building large-scale networked systems. Professor Xu now leads a collaborative research team at Leeds studying Internet and Cloud technologies with a focus on complex system engineering, system security and dependability, and evolving system architectures. He is the recipient of the BCS/IEE Brendan Murphy Prize 2001 for the best work in the area of distributed systems and networks. He has led or co-led many key research projects, and published more than 280 academic papers and edited books. Professor Xu has served as general Chair/Program Chair/PC member of numerous international computer conferences and has been an editorial board member for several international Computing journals.

Dr Dacheng Zhang is a senior researcher in Beijing Research institute of Huawei Technologies Co. Ltd, China. He obtained a PhD degree from the School of Computing, the University of Leeds, MSc Degree in Computer Science from the University of Durham, and BSc. degree in Computer Science at Northern Jiaotong University. His research area covers security issues with Web services, Database Transactions, and security issues with routing protocols. He has directed multiple research projects in corporation with different universities. He has also engaged in multiple important research projects founded by Chinese government (e.g. Chinese National 863 Program) as a principal researcher. He is active in the Internet Engineering Task Force. His interests in standardization work include the next generation of routing architectures, mobile IP, and IPv6.

Dr Lu Liu is Senior Lecturer in School of Computing and Mathematics, University of Derby (UK). Before joining University of Derby, he was Lecturer in School of Engineering and Information Sciences at Middlesex University (UK). Prior to his academic career, he was Research Fellow in the School of Computing at the University of Leeds (UK), working on NECTISE Project which was an UK EPSRC/BAE Systems funded research project involving ten UK Universities and CoLaB Project which was funded by UK EPSRC and China 863 Program. He received a Ph.D degree (funded by UK DIF DTC) from the University of Surrey (UK) and M.Sc. degree from Brunel University (UK). His research interests are in areas of peer-to-peer computing, software engineering and service-oriented computing. Dr Liu has over 30 scientific publications in reputable journals, academic books and international conferences. He won the Best Paper Award at the Realising Network Enabled Capability Conference in 2008. He is on the Editorial Review Board of International Journal of Distributed Systems and Technologies. He served as Co-chair, TPC Member, Advisory Committee Member and Session Chair of many international conferences and workshops. He is a member of IEEE.

Professor Xianxian Li is a professor at the Institute of Advanced Computing Technology, Beihang University, China. He received the Master degree in Mathematics from Guangxi Normal University in 1997, and received the Ph.D degree in Computer Science and Technology from Beihang University. As a principal researcher and manager he engaged in many important projects like National Natural Science Foundation, National 863 Program and National Defense Foundation etc. He obtained the Second-class China State Science and Technology Award in 2004 and the first prize of National Defense Science and Technology.