A Security Framework for Distributed Brokering Systems

Shrideep Pallickara[1], Marlon Pierce[1], Geoffrey Fox[1,2], Yan Yan[1,2], Yi Huang[1,2]
{spallick, marpierc, gcf, yayan, yihuan}@indiana.edu
Community Grid Computing Labs, Indiana University[1]
Department of Computer Science, Indiana University[2]

Contact:
Shrideep Pallickara
501 N. Morton St., Suite 224
Bloomington, IN-47408
spallick@indiana.edu

Abstract

Loosely coupled, globally scalable distributed systems, including both peer-to-peer systems and computational grids, rely on the transmission of messages and events that may transverse many point-to-point connections and may need to reach several destinations. The identity of entities, the authorization to send or receive certain messages, and the privacy and integrity of those messages must all be established. In this paper we present a system design that addresses the security requirements for messaging systems that employ the generalized topic-based publish/subscribe paradigm. In particular, we address initial authentication and maintenance of identity, scalable topic security, and message-level security that protects messages over multiple hops with varying underlying transport security. We also review several potential forms of attacks on the system and the steps we take to thwart such attacks.

Student Status: Yan Yan and Yi Huang are full time graduate students.
Submission for REGULAR PRESENTATION

A Security Framework for Distributed Brokering Systems

Shrideep Pallickara[1], Marlon Pierce[1], Geoffrey Fox[1,2], Yan Yan[1,2], Yi Huang[1,2]
{spallick, marpierc, gcf, yayan, yihuan}@indiana.edu
Community Grid Computing Labs, Indiana University[1]
Department of Computer Science, Indiana University[2]

Abstract

Loosely coupled, globally scalable distributed systems, including both peer-to-peer systems and computational grids, rely on the transmission of messages and events that may transverse many point-to-point connections and may need to reach several destinations. The identity of entities, the authorization to send or receive certain messages, and the privacy and integrity of those messages must all be established. In this paper we present a system design that addresses the security requirements for messaging systems that employ the generalized topic-based publish/subscribe paradigm. In particular, we address initial authentication and maintenance of identity, scalable topic security, and message-level security that protects messages over multiple hops with varying underlying transport security. We also review several potential forms of attacks on the system and the steps we take to thwart such attacks.

1.0 Introduction

The Internet is presently being used to support increasingly complex interaction models as a result of more and more applications, services and frameworks becoming network centric. Applications and services interact with devices which span a very wide spectrum that includes desktops, PDAs and other handheld devices, appliances, and other networked resources. Systems that currently proliferate on the Internet include enterprise middleware systems, peer-to-peer (P2P) systems, grid systems and Web Services based systems. Entities within these systems communicate with each other through exchange of messages. These messages encapsulate information pertaining to transactions, search, discovery and subsequent sharing of resources, exposing and utilizing resources, etc. These entities would not necessarily be part of the same domain or within the same local area network.

To manage the volume of entities, the messaging infrastructure that processes these interactions are usually hosted on a network of cooperating messaging nodes, which we call broker nodes. The processing and servicing of interactions is in itself a distributed problem that involves several nodes and the links that connect them. Increasingly, messages issued by an entity are routed to the entities that the initiating entity is not directly aware of (the messaging infrastructure computes the destinations). Entities may wish to communicate with each other while ensuring that the content can be viewed only by authorized entities. Interactions, as they traverse through the messaging infrastructure, traverse over several node hops prior to receipt at an interested entity. Interactions thus may need to traverse over firewalls, proxies and NAT boundaries, some of which may prevent creation of secure communication links between two nodes within the system. Communications between entities will thus entail communications over insecure links.

In this paper we propose a scheme to ensure secure communications between authorized entities in a distributed brokering system that supports the generalized publish/subscribe paradigm. The scheme should also incorporate strategies to detect a security compromise while reducing vulnerability to certain kinds of attacks. We investigate these issues in the context of our advanced research prototype, NaradaBrokering [1-9]. NaradaBrokering provides support for centralized, distributed, and P2P [10] interactions. The generalized publish/subscribe framework involves entities specifying an interest in a certain topic. Publishers publish messages to a given topic and upon receipt of these messages the system computes the destinations (subscribers) that should receive these messages.

In our approach we secure messages independently of any transport level security. This provides a fine-grained security structure suitable for distributed systems and multiple security roles. For example, parts of the message may be encrypted differently, allowing users with different access privileges to access different parts of the message. Basic security operations such as authentication should be performed in a mechanism-independent way, with specific mechanisms (Kerberos [11], PKI [12]) plugged into specific applications. The message level security framework allows us to deploy communication links where data is not encrypted. Furthermore, this scheme also ensures that no node/unauthorized-entity ever sees the unencrypted message. In our strategy we incorporate schemes to detect and respond to security compromises while also dealing with various attack scenarios.

2.0 Related work

GKMP [13] outlines an architecture for the management of cryptographic keys for multicast communications. The GKMP creates key for cryptographic groups and distributes this key securely to the group members while incorporating peer review to incorporate the security policy. GKMP also denies access to known compromised hosts, while monitoring permissions and updating them. In [14] strategies for reducing the number of encryptions required to preserve confidentiality between an end-point broker and its subscribing entities in the context of Content based publish-subscribe systems.

P2P systems incorporate several strategies that address secure interchange while incorporating strategies to incorporate trust and reputations. Groove [15] provides excellent P2P security by securing shared spaces, which comprise documents, messages etc. Incremental changes to a shared space object are transmitted to authorized peers in a secure way. Systems such as http://www.advagato.org incorporate trust metrics to support reputations while defeating scenarios where users band together to boost reputation scores. The Free Haven system [16] provides strategies for incorporating accountability while maintaining peer anonymity. Each server in Free Haven maintains values pertaining to reputation and credibility, while broadcasting referrals in some cases.

The Grid Security Infrastructure (GSI) [17] provides a complementary approach that addresses a related problem: a user may need to invoke a particular service through one or more proxy servers. GSI breaks this request into a chain of point-to-point invocations, with the user's initial (proxy) credential used to create a sequence of proxy key pairs. Each key pair is delegated limited authority to invoke a remote service. Thus the GSI approach treats secure end-to-end connections as a sequence of secure point-to-point connections. We take a complementary approach that enforces security at the endpoints and allows the message to travel securely through insecure intermediaries. The Akenti system [18] addresses the important problem of authorization of resources in a distributed system with multiple stakeholders. Akenti provides an XML access policy language that is transmitted using X.509 policy certificates. This system is complementary to the authentication and message privacy issues that we concentrate on and could potentially be used to govern access to publishing topics. Legion (http://www.cs.virginia.edu/~legion/) is a long-standing research project for building a "virtual computer" out of distributed objects running on various computing resources. Legion objects communicate within a secure messaging framework [19] with an abstract authentication/identity system that may use either PKI or Kerberos. Legion also defines an access control policy on objects.

There are many emerging issues pertaining to security in XML-based Web Services. WS-Security [20] from IBM and Microsoft outlines a proposed architecture to address the gaps between existing security standards and Web Services such as SOAP [21]. By abstracting security services, the WS-security model also serves to unify security technologies such as PKI and Kerberos. Security specifications for Web Services are just starting to emerge, but generally follow the same approach: the message creator adds a signed XML message containing security statements to the SOAP envelope. The message consumer must be able to check these statements and the associated signature before deciding if it can execute the request. Web Services such as those based on SOAP are essentially exchanging XML messages. XML-based message-level security has the additional advantages that it builds upon existing specifications for signing (XML signatures) [22] and encryption (XML encryption) [23], and also allows us to develop a basic security package that can work with both Web Services (communicating with SOAP) as well as peers. The Security Assertion Markup Language (SAML) [24] from OASIS deals with the standard representation of security data – authentication, authorization and attribute – which would be recognized by different application security services irrespective of the security technology or policy that the deploy. SAML is designed to work with W3C specifications such as XML Signature and SOAP. XKMS (XML Key Management Specification) [25] specifies the language for key based trust services and includes protocols for registering, locating and validating keys. Finally, XACML (XML Access Control Markup Language) [26] specifies a vocabulary for expressing XML-formatted rules for making authentication and authorization assertions. XACML uses SAML to define subjects and associated actions.

The Open Grids Services Architecture (OGSA) extends the Web Service Description Language to address necessary issues such as service metadata and service state. OGSA services, like Web Services, ultimately comprise a message-based architecture. Client request messages may be encoded for example in SOAP and passed to hosting environments for consumption and execution. The security issues of this system have been reviewed in [27]. Besides client-server-service style invocations, OGSA proposes to define an interface language that would allow services to communicate their state changes with each other through a notification framework. This notification scheme would in practice be bound to various messaging implementations, such as NaradaBrokering. The security scheme described herein may be used to secure the service-to-service messaging layer.

3.0 The Security System Architecture

The basic infrastructure comprises of the broker network and the Key Management Center (KMC) (see Figure 1). For the purposes of our discussion we assume that there is only one KMC within the system. Extending this basic scheme to include multiple KMCs, residing in different domains, will be discussed in section 6.0. The KMC provides a host of functions specific to the management of keys within the system while also incorporating an authorization module, which it uses to keep track of authorizations that different entities within the system possess. The functions performed by the KMC include the management of keys associated with entities and topics, while ensuring secure communications with the entities. Entities use SSL for communications with the KMC. All entities in the system possess a public/private key pair. Entities register their public keys with the KMC.

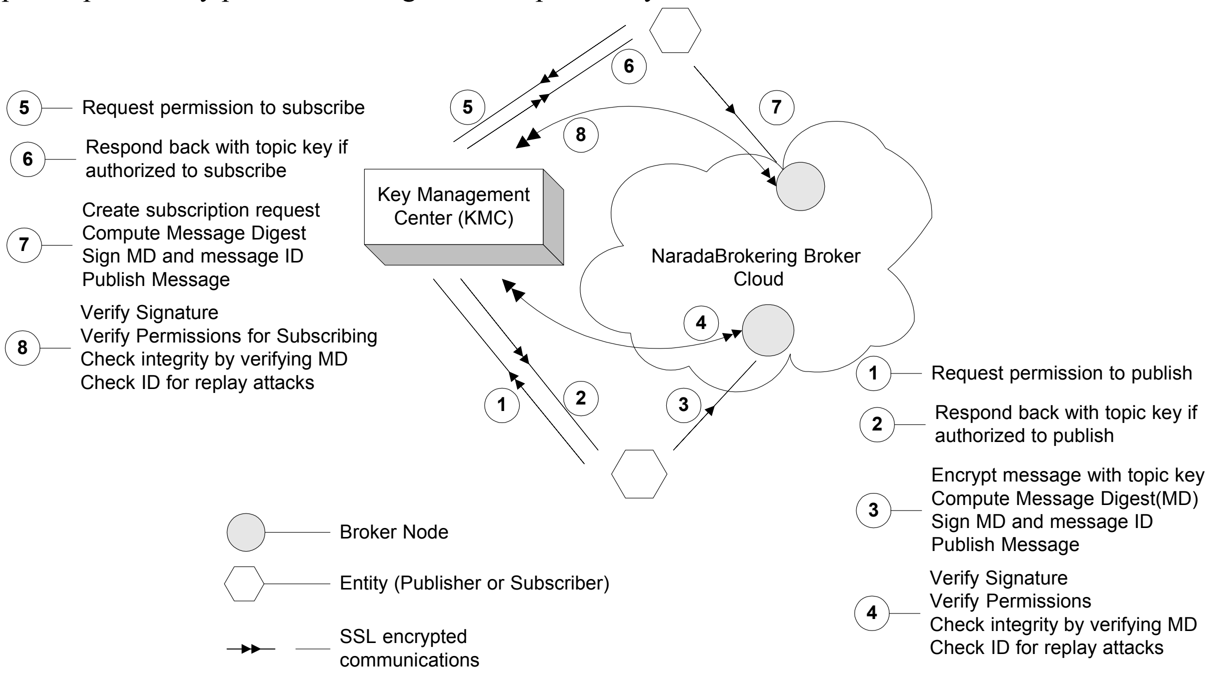

5 Request permission to subscribe
6 Respond back with topic key if authorized to subscribe
7 Create subscription request
Compute Message Digest
Sign MD and message ID
Publish Message
8 Verify Signature
Verify Permissions for Subscribing
Check integrity by verifying MD
Check ID for replay attacks
Key Management Center (KMC)
NaradaBrokering Broker Cloud
1 Request permission to publish
2 Respond back with topic key if authorized to publish
3 Encrypt message with topic key
Compute Message Digest(MD)
Sign MD and message ID
Publish Message
4 Verify Signature
Verify Permissions
Check integrity by verifying MD
Check ID for replay attacks
Broker Node
Entity (Publisher or Subscriber)
SSL encrypted communications

Figure 1: Framework for secure messaging

The secure messaging scheme has two basic parts: initial authentication (proof of identity) by all publishers and subscribers, followed by validated, secured publishing and receiving. In the initial authentication step, a publisher or subscriber would send its request to the system, signed with its private key. The request message's signature can then be verified. The decision to allow the entity to publish or subscribe to a particular topic is determined from access control lists. Associated with every topic is an Access Control List (ACL) identifying entities that are authorized to subscribe to messages published to that topic. A similar ACL exists for publishers. When an entity indicates an interest in publishing/subscribing to a topic, and once it clears the authorization process, depending on the strategy used to achieve secure messaging, it could be returned a topic-key encrypted with the entity's public personal-key. Brokers within the broker network are also involved in determining if the publisher is indeed authorized to publish messages.

3.1 Authorized Publishing and Subscribing

Entities use the broker that they are connected to, to funnel interactions to the brokering system. These interactions include publishing messages to a given topic and subscribing to a certain topic. When an entity seeks to publish/subscribe to a topic, it issues a request to the KMC. If the entity is authorized to publish/subscribe to this topic the KMC returns the relevant topic key, encrypted with the entity's public key. Interactions initiated by entities with the brokers need to include information which allow individual brokers to verify if the interaction is an authorized one and also to detect if the message has been tampered with. Message digests provide an indication that the interaction encapsulated within the message was not tampered en route to its destinations. A malicious user can exploit vulnerabilities in collisions arising from the hashing function employed to compute the digest. Having a larger digest increases the integrity of the message. MD5 [28] generates a 128-bit message digest, while SHA-1 [29], generates a 160-bit value. In order to allow the broker to identify and verify the source of the message, entities sign the interactions that they funnel into the broker network.

Subscriptions are propagated to relevant parts of the system to ensure that messages published to the subscribed topic are routed to the subscribed entities. The entity now proceeds to propagate its subscription within the broker network. It does this by signing its subscription request and the unique ID associated with its subscription request, with its private key. Each broker that encounters this subscription propagation can verify the signature and whether the entity is authorized to subscribe to the topic.

Every secure message contains the topic that it being issued for and a signature of the publisher. When a message is ready to be issued, the publisher signs the encrypted (based on one of 3 encryption schemes that we outline in the subsequent section) payload by encrypting the computed message digest of the payload with its private key. This allows nodes to verify the authenticity of the published message and ensures that messages published by unauthorized publishers are not routed to the subscribers. Brokers en route to the final destinations can also verify this signature to test the source.

When messages are being routed through the broker network individual brokers can verify if the signing publisher is indeed authorized to publish to this topic. Since there could be multiple publishers to a given topic, individual brokers keep track of the authorized list of publishers to a given topic. Individual brokers do the authorization confirmation the first time they receive a message from a publisher. This confirmation is done in tandem with the Authorization module existing within the KMC. Once the signature is verified, the broker proceeds to route the message. Figure 1 depicts the sequence of operations that we outlined in this section.

3.2 Secure Delivery of Messages

There are three different strategies that can be deployed for secure messaging within the system. Depending on the strategy deployed for secure messaging there could be encryption keys associated with topics. Furthermore, the type of keys (symmetric or asymmetric) associated with individual topics also varies. Choice of the strategy, to be used for secure messaging, is within the purview of the topic creator.

3.2.1 Secure messaging based on personal keys

In this section we discuss issues involved in doing secure messaging using the personal public key of the entities. This approach presents some trade-offs in terms of security and performance. In this approach since we have the pubic keys for every subscriber, we can encrypt (done by publisher) every message to a particular topic with each of the subscriber's public keys. This ensures the message security and obeys the usual PKI restriction that no private keys ever be exchanged. This however requires N duplicate messages to be created and individually signed, where N is the number of subscribers. This would probably introduce unacceptable performance degradation. This also prevents decoupled communications where a publisher needs to be aware of every potential subscriber.

3.2.2 Secure messaging based on asymmetric topic key pairs

In this approach when a topic is created, there is an asymmetric topic-key pair associated with the topic. The public topic-key associated with the topic is used by authorized publishers to encrypt the message contents while the private topic-key is used by authorized subscribers to decrypt the encrypted message. Upon completion of the authentication process and depending on their authorizations the relevant topic-key – public, private or both – is delivered securely to the relevant entities by encrypting them with the entity's public personal-key. Individual entities can decrypt this topic key(s) with their private personal-key. Compared to the secure messaging based on personal keys this scheme obviates the need for multiple encryptions. In this approach the publisher of a message encrypts the message only once with the public topic-key. The securely distributed private topic-key is then used by authorized subscribers to decrypt the message contents. This approach is much more efficient but carries the risk that each subscriber must maintain the security of the private topic-key. If any of the N authorized subscribers loses secure control of its copy of the private topic-key, the entire topic becomes unsecured. Assigning short life times to the topic keys and renewing them frequently can mitigate this problem.

3.2.3 Secure messaging based on symmetric topic keys

Secure messaging based on asymmetric topic keys reduces the number of encryptions need in the approach based on personal keys. Since approaches outlined in sections 3.2.1 and 3.2.2 rely on using asymmetric encryptions to secure message payload, they inherit problems concomitant with the encryption scheme. Encryptions based on asymmetric keys tend to be more expensive (100 to 1,000 times slower) than the ones based on symmetric keys. Depending on the type of applications the costs would end up being very prohibitive. In the secure messaging scheme based on symmetric keys [30], there is only a symmetric key associated with a topic. This topic-key is distributed securely to authorized publishing/subscribing entities by encrypting it with the each authorized entity's public personal-key.

3.2.4 Issues pertaining to topic keys

The architecture also needs to provide a suite of ciphers for encryptions. Trade offs between encryption strength and the performance of the encryption algorithms need to be considered while determining the key lengths for encryptions. Having topic keys associated with topics also enables content providers to charge for content. Topic keys could be distributed for a certain charge and could also have lifetimes associated it to ensure that entities do not use services without first paying for them. Short key lifetimes in general tend to mitigate the effects of lost/stolen keys. The guaranteed delivery properties within the system could be used to maintain audit trails within the system. Furthermore, it is also conceivable that a given message sent to a topic could have different parts encrypted using different keys. Different keys would have different premiums associated with them.

4.0 Dealing with various attack scenarios

In this section we outline the various attack scenarios that we try to deal with. We do not address (and consider it out of our research scope) cryptographic attacks. The cryptographic packages we use include IAIK [31] and Sun's JCE [32].

Man-in-the-middle attacks

Man-in-the-middle (MITM) attacks involve an attacker intercepting and replacing public keys of two communication parties with its own public key. This allows the attacker to decrypt communications using his/her private key. The initial topic key exchanges between the entities and the KMC are vulnerable to this kind of attack. We solve this by requiring that all communications with between the entities be over SSL, which eliminates MITM attacks. MITM attacks are not a problem for message transmission, since topic keys have already been exchanged over SSL and individual messages are encrypted and signed.

Replay attacks

In replay attacks the attacker stores network packets and resends them at a later time. SSL/TLS defeats this during communications between entities and the KMC. For entities communicating with each other, through the messaging infrastructure, each message in the system has a unique ID associated with it. The publisher would sign both the digest of the payload and the ID. Messages with the same message-ID will be garbage collected at individual brokers thus preventing the broker network from expending network and CPU cycles on processing the replayed message [9].

Denial of service

In denial of service attacks the attacker may try to overload the system resources such as CPU and network cycles by generating a large volume of spurious messages that are processed by the system. Since only authorized entities are allowed to publish messages within the system, messages published by unauthorized entities would be rejected at brokers that receive them. The KMC may be vulnerable to multiple bogus requests originating from a malicious entity. This particular vulnerability may be addressed in the implementation by rejecting socket connections from IP addresses that have made multiple bogus attempts. Distributed systems by their nature generally tend to be less susceptible to denial of service attacks.

Dealing with rouge brokers

In our scheme individual brokers route the encrypted messages based on their topic headers. It is possible that a malicious broker may randomly drop messages. This is dealt with in two ways. First, messages can take multiple routes to reach their destinations. Numbering information in these messages along with information pertaining to failed brokers could be used to identify rouge brokers. The broker network can then reorganize its connections to the detected rouge broker. Entities attached to the rouge broker could either be induced to relocate to another broker or they would eventually relocate to another broker due to prolonged periods of inactivity or incorrect inactivity (as in message replays etc.) Second, it is also possible to develop a broker-to-broker layer of security. Here, each broker verifies the other brokers that it is communicating with. In PKI this can be done with the usual encryption-plus-signature scheme. This introduces additional performance overhead, but can be used to prevent or detect the presence of rogue brokers.

Non-repudiation

This is more of a system abuse than an attack. For example, a user publishes something malicious, then throws away his key and claims never to have sent the malicious message. The user may claim that the key was never really delivered to it. The user may also claim that someone stole the topic-key during the key transmission and used it. This is defeated by SSL and mutual authentication in the transport layer during the initial key distribution phase. Another abuse is that the user does something malicious and then claims his private key was stolen (perhaps delivering it to some other user or anonymously posting it on a public web site). Protections against this are the same as for "legitimately" stolen keys, which we discuss in the subsequent section.

5.0 Detecting and responding to a security compromise

One of the ways to detect security compromises is to issue authentication challenges at regular intervals along with shorter key lifetimes. System topics, which effectively define the services, may have unpredictable lifetimes ranging from seconds to years. Most current systems are designed for fixed user session periods (a few hours) or for long-term key lifetimes (a year or two). Entities would need to negotiate and retrieve new keys after the delivery of a set of messages or a period of time. Additionally entities may be forced to answer queries from a set negotiated between the KMC and the entity during initializations.

When it is detected or reported that an entity's security has been compromised the following are the operations that need to be performed –

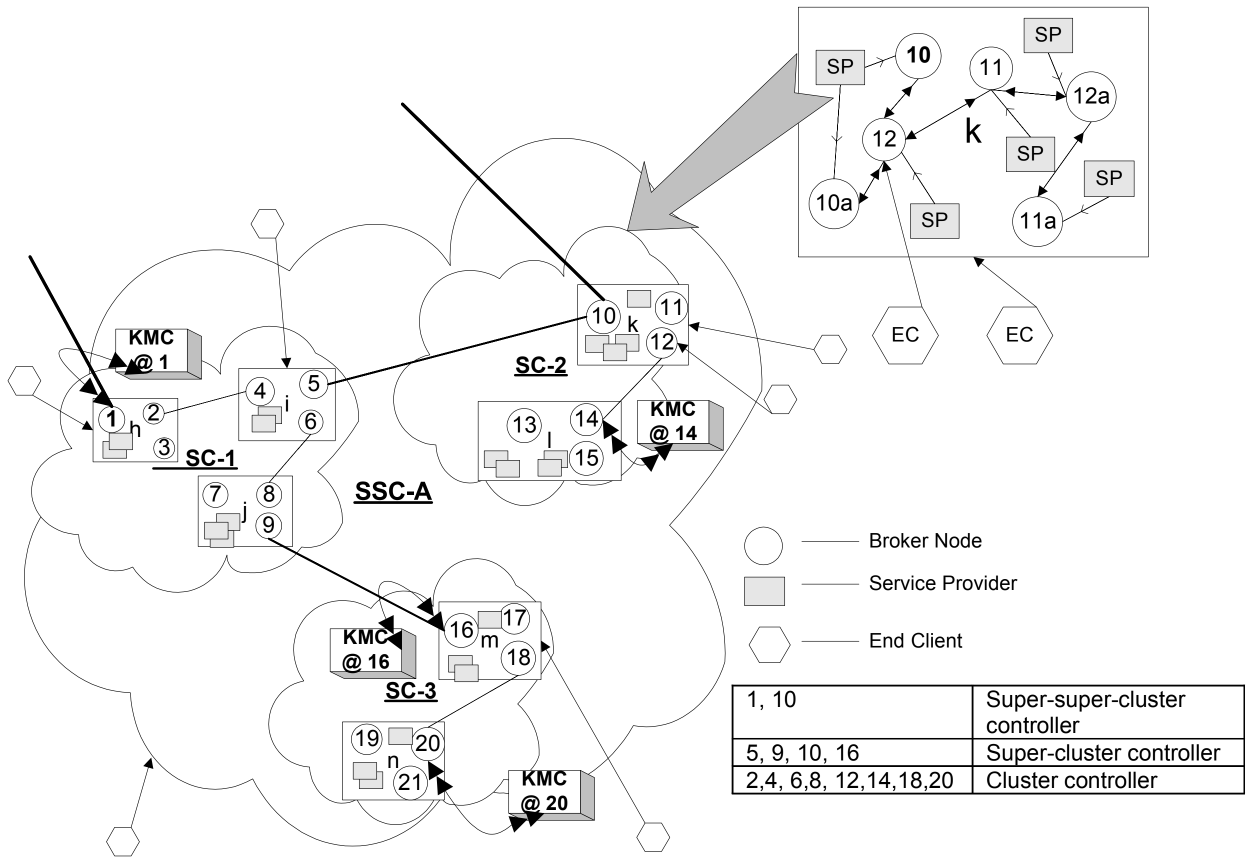
Generation of new keys: New keys need to be generated for the topics that the entity can publish and subscribe to.

Propagation of compromise detection: A message also needs to be propagated throughout the system (to brokers and entities alike) propagating the invalidity of the affected entity's signature. Entities (currently present in the system) that receive these notifications and are affected by it, renegotiate new keys for the affected topics. We could require disconnected entities to do a check on whether the keys for any of the topics that it publishes/subscribes to have changed. If it has it needs to retrieve these new topic keys.

Encrypting replay of messages with new keys: Routing of missed messages is done based on the new key.

6.0 Distributed Key Management Centers

To address the issues of scaling, load balancing and failure resiliency, NaradaBrokering is implemented on a network of cooperating brokers.



1, 10	Super-super-cluster controller
5, 9, 10, 16	Super-cluster controller
2,4, 6,8, 12,14,18,20	Cluster controller

Figure 2:An example of a NaradaBrokering broker network sub-section with multiple KMCs.

In NaradaBrokering we impose a hierarchical structure on the broker network, where a broker is part of a cluster that is part of a super-cluster, which in turn is part of a super-super-cluster and so on. Figure 2 depicts a sub-system comprising of a super-super-cluster **SSC-A** with 3 super-clusters **SC-1**, **SC-2** and **SC-3** each of which have clusters that in turn are comprised of broker nodes. Clusters comprise strongly connected brokers with multiple links to brokers in other clusters, ensuring alternate communication routes during failures. This organization scheme results in "small world networks" [33,34] where the average communication pathlengths between brokers increase logarithmically with geometric increases in network size, as opposed to exponential increases in uncontrolled settings. This distributed cluster architecture allows NaradaBrokering to support large heterogeneous client configurations that scale to arbitrary size. To review briefly units (super-super-clusters, super-clusters, clusters) comprise multiple sub-units (super-clusters, clusters, broker nodes). Also, within every unit, there is at least one unit-controller, responsible for facilitating communications with nodes in other units. For e.g. in figure 2 cluster controller node **20** provides a gateway to broker nodes in cluster **m**.

We follow a similar hierarchical structure in our organization of KMCs, as depicted in figure 3. Each broker within a unit is within the scope of the KMC managing that unit. By a KMC's scope we mean that clients connected to brokers, within a given unit, delegate the management of keys and ACLs associated with newly created topics to the KMC in question. Figure 3 depicts the broker nodes scoped by individual KMCs. In most practical situations all users within a given domain would negotiate or interact with the KMC managing its domain. There could of course be multiple KMCs within a given domain. KMC's are hosted at nodes that are unit controllers and within a given unit there can be only one KMC that is responsible for managing that unit. Thus, within a cluster there can be only one KMC that scopes the cluster irrespective of the number of cluster controllers in that cluster. It is, however, possible that there are KMCs responsible for managing the lower units. Similar to the X.509 [35] Certificate chaining a KMC immediately higher up in the KMC chain can verify a given KMC's signature. Setting up of a KMC within the KMC chain requires authorizations from the KMC one link above and KMCs one link below in the KMC chain. In figure 3**,** if **KMC@16** were to be set up *after* **KMC@1** and **KMC@20** *are present* in the system, **KMC@16** would require authorizations from both **KMC@1** and **KMC@20**. Furthermore, **KMC@20** would now have its signature verified by **KMC@16**. If a new **KMC@18** were to be set up at cluster controller node **18**, nodes **16, 17, 18** would be scoped by this new KMC instead of **KMC@16**.

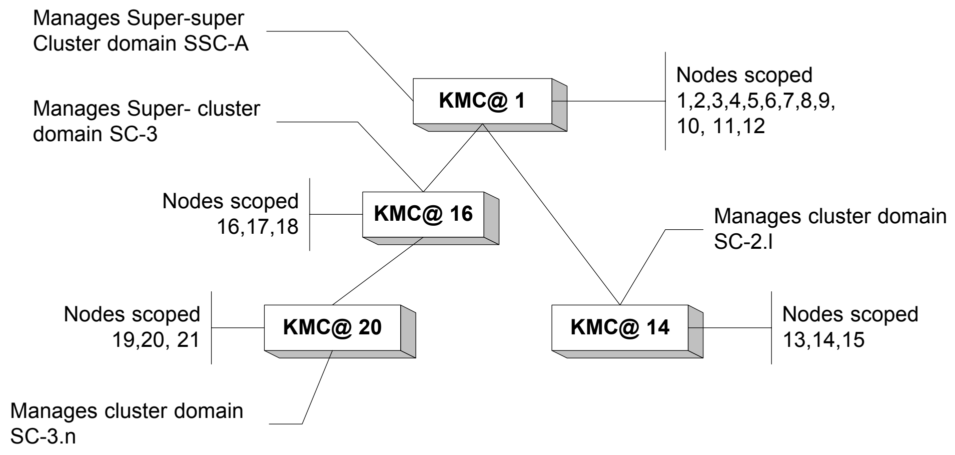

Manages Super-super Cluster domain SSC-A
KMC@ 1
Nodes scoped 1,2,3,4,5,6,7,8,9, 10, 11,12
Manages Super- cluster domain SC-3
Nodes scoped 16,17,18
KMC@ 16
Manages cluster domain SC-2.l
Nodes scoped 19,20, 21
KMC@ 20
KMC@ 14
Nodes scoped 13,14,15
Manages cluster domain SC-3.n

Figure 3: The KMC hierarchy

While processing requests originating from entities outside its managed scope, a KMC processes only those requests that have been signed by a KMC higher up in the KMC domain that both the entity and the processing KMC are aware of. Thus, **KMC@14** will process requests from entities in cluster **SC-3.n** only if the request is signed by **KMC@1**. ACL's and keys associated with a topic are maintained at the

KMC at which the topic was first created and registered. This KMC also has the final say in accepting authorizations to topics under its purview.

Presence of topics is propagated up the KMC chain until it reaches the highest level. Only the topics and information pertaining to the KMC that stores the topic key is routed to the higher level KMCs. When trying to create a topic, a request is propagated up the KMC chain to see if the topic exists. If it does not, the topic and relevant keys are created and an ACL which includes the creator as a publisher and subscriber to the topic is also created. Authentication challenges (done at regular intervals to detect security compromise) to clients are issued by the KMC at which the topic keys and associated ACL information are maintained. KMCs at the root maintain information about all the registered topics within the system. Associated with topics each KMC maintains the following information

a) Keys, symmetric or asymmetric ones.
b) Access Control information and alternate authentication challenges/queries for authorized registered users. These are used in detecting security compromises.
c) In case the topic in question is not managed by a given KMC, items (a) and (b) are not part of the information that is maintained. Instead, we include information pertaining to the KMC that manages the topic and also the unit that this KMC is a part of.

For authentication and authorization purposes when an entity connects from another domain they provide information pertaining to the KMC that can authorize them. Thus, subscriptions contain the subscribing client's signature and also information pertaining to the KMC that can verify this signature. A given broker verifies if this KMC is a trusted one and then proceeds to process the request accordingly. This information is stored in the trust store which is updated periodically to eliminate keys/signatures that are old. Topic key compromises are dealt with the KMC managing the topic, while entity key compromises are propagated to relevant parts of the system.

7.0 Future work

We intend to investigate issues pertaining to security in the context of search and discovery of resources in P2P systems and audio/video conferencing in NaradaBrokering. Another area of research is the incorporation of trust metrics and management of reputations within the system. A body of work in this area exists in the P2P domain and it would be interesting to investigate these issues in the context of distributed brokering and grid systems.

8.0 Conclusion

In this paper we presented a strategy to secure messages exchanged between entities. Communications between these entities may take place over insecure links. The scheme provides a framework for achieving end-to-end integrity while ensuring that authorized entities are the only ones that publish, subscribe, and decrypt messages sent to a topic. We have implemented a prototype of the security strategy, presented in this paper, in the context of a centralized KMC. We intend to implement the distributed KMC scheme that was outlined in this paper. The final version of this paper will include comprehensive results from our ongoing implementations.

References

1. The NaradaBrokering System http://www.naradabrokering.org
2. NaradaBrokering: An Event Based Infrastructure for Building Scaleable Durable Peer-to-Peer Grids. Geoffrey Fox and Shrideep Pallickara. Chapter 22 of "Grid Computing: Making the Global Infrastructure a Reality". John Wiley April'03.
3. The Narada Event Brokering System: Overview and Extensions. Geoffrey Fox and Shrideep Pallickara. Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, June 2002. pp 353-359.
4. A Scaleable Event Infrastructure for Peer to Peer Grids. Geoffrey Fox, Shrideep Pallickara and Xi Rao. Proceedings of ACM Java Grande ISCOPE Conference 2002. Seattle, Washington. November 2002.
5. "JMS Compliance in the Narada Event Brokering System." Geoffrey Fox and Shrideep Pallickara. Proceedings of the International Conference on Internet Computing (IC-02). June 2002. pp 391-402.
6. Grid Services for Earthquake Science. Fox et al. Concurrency & Computation: Practice & Experience.14(6-7):371-393.

7. "Integration of NaradaBrokering and Audio/Video Conferencing as a Web Service". Hasan Bulut, Geoffrey Fox, Shrideep Pallickara, Ahmet Uyar and Wenjun Wu. Proceedings of the IASTED International Conference on Communications, Internet, and Information Technology, November, 2002, in St.Thomas, US Virgin Islands.
8. "An Approach to High Performance Distributed Web Brokering". Fox and Pallickara, ACM Ubiquity 2:38. Nov 2001.
9. An Event Service to Support Grid Computational Environments Geoffrey Fox and Shrideep Pallickara. Journal of *Concurrency and Computation: Practice & Experience*. Volume 14(13-15) pp 1097-1129.
10. Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology. Edited by Andy Oram. O'Rielly Press, CA. March 2001.
11. "Kerberos: An Authentication Service For Open Networked Systems". J. Steiner, C. Neuman, and J. Schiller. In Proceedings of the Winter 1988 USENIX Conference, February 1988.
12. "Applied Cryptography," B. Schneier. John Wiley and Sons. New York, 1996.
13. H. Harney and C. Muckenhirn. Group Key Management Protocol (GKMP) Specification. IETF RFC 2093. July 97.
14. L. Opyrchal and A. Prakash. "Secure Distribution of Events in Content-Based Publish Subscribe Systems." In Proceedings of the 10th USENIX Security Symposium, pages 281--295, August 2001.
15. Groove Networks Inc. Desktop Collaboration Software. http://www.groove.net/
16. Roger Dingledine, Michael J. Freedman, David Hopwood, David Molnar. A Reputation System to Increase MIX-net Reliability. Proceedings, Information Hiding Workshop, Mar 2001 (LNCS 2137).
17. A Security Architecture for Computational Grids," I. Foster, C. Kesselman, G. Tsudik, S. Tuecke. Proc. 5th ACM Conference on Computer and Communications Security Conference, pp. 83-92, 1998
18. "Certificate-based Access Control for Widely Distributed Resources" Mary Thompson, William Johnston, Srilekha Mudumbai, Gary Hoo, Keith Jackson, Proceedings of the Eighth Usenix Security Symposium, Aug. `99.
19. "A Flexible Security System for Metacomputing Environments" (HPCN Europe 99), April 1999.
20. "Web Services Security (WS-Security) Version 1.0 05 April 2002," B.Atkinson, et al. Available from http://www-106.ibm.com/developerworks/webservices/library/ws-secure/.
21. XML based messaging and protocol specifications SOAP. http://www.w3.org/2000/xp/.
22. "XML-Signature Syntax and Processing, W3C Recommendation 12 February 2002", M. Bartel, J. Boyer,B Fox, et. al. Available from http://www.w3.org/TR/xmldsig-core/
23. "XML Encryption Syntax and Processing, W3C Recommendation 10 December 2002", T. Imamura, B. Dillaway, E. Simon Available from http://www.w3.org/TR/xmlenc-core/
24. "Assertions and Protocol for the OASIS Security Assertion Markup Language," P. Hallam-Baker and E. Maler, eds. Available from http://www.oasis-open.org/ committees/security/docs/ cs-sstc-core-01.pdf.
25. "XML Key Management Specification (XKMS 2.0), W3C Working Draft 18 March 2002", Edited by P Hallam-Baker, Available from http://www.w3.org/TR/xkms2/
26. "OASIS eXtensible Access Control Markup Language (XACML)" edited by S. Godik, T. Moses, Available from http://www.oasis-open.org/committees/xacml/repository/cs-xacml-specification-01.pdf
27. http://www.globus.org/ogsa/Security/OGSA-SecArch-v1-07192002.pdf (Global Grid forum document).
28. The MD5 Message-Digest Algorithm. R. Rivest. Network Working Group. Internet RFC 1321.
29. The Secure Hash Algorithm (SHA-1) specified in FIPS 180-1. Available from http://www.itl.nist.gov/fipspubs/fip180-1.htm
30. "AES Proposal: Rijndael", J. Daemen, V. Rijmen, Available fromhttp://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael.pdf
31. The IAIK Java Cryptography Extension (IAIK-JCE) toolkit. http://jce.iaik.tugraz.at/products/01_jce/documentation/javadoc
32. The JavaTM Cryptography Extension (JCE). Available from http://java.sun.com/products/jce/
33. "Collective Dynamics of Small-World Networks". D.J. Watts and S.H. Strogatz. Nature. 393:440. 1998.
34. "Diameter of the World Wide Web". R. Albert, H. Jeong and A. Barabasi. Nature 401:130. 1999.
35. Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework. S. Chokhani and W. Ford, RFC 2527. March 1999