A Framework for Secure End-to-End Delivery of Messages in Publish/Subscribe Systems

Shrideep Pallickara, Marlon Pierce, Harshawardhan Gadgil, Geoffrey Fox, Yan Yan, Yi Huang

Community Grids Lab, Indiana University
501 N Morton St, Suite 224
Bloomington, Indiana 47401- USA
spallick@indiana.edu
marpierc@indiana.edu
hgadgil@indiana.edu
gcf@indiana.edu
yayan@indiana.edu
yihuan@indiana.edu

***Abstract*— In this paper we present a framework for the secure end-to-end delivery of messages in distributed messaging infrastructures based on the publish/subscribe paradigm. The framework enables authorized publishing and consumption of messages. Brokers, which constitute individual nodes within the messaging infrastructure, also ensure that the dissemination of content is enabled only for authorized entities. The framework includes strategies to cope with attack scenarios such as denial of service attacks and replay attacks. Finally, we include experimental results from our implementation.**

I. INTRODUCTION

Entities in distributed systems communicate through the exchange of messages. The underlying messaging framework in these systems could be based on point-to-point communications, queuing systems, peer-to-peer (P2P) based interactions, hardware multicast-based disseminations or publish/subscribe systems. This work focuses on publish/subscribe systems.

In publish/subscribe systems typically there are one or more router nodes, referred to as *brokers*, which are responsible for routing messages from the publishers to the subscribers. Individual messages are routed based on the content descriptor, referred to as the *topic*, information contained within these messages. This information is added by the publisher to describe the message contents. Subscribing entities need to first register their interests in specific topics – the *subscriptions* – with the broker.

Depending on the expressiveness of the content description, there are two kinds of topics viz. simple and complex. In the case of *simple topics*, this is usually a "/" separated String such as /Sports/Football. *Complex topics* describe the content in a more verbose manner. Here, the content could be described using a set of tag=value pairs, a set of properties associated with the message, verbose text or as an XML (eXtensible Markup Language) document. In each of these cases the corresponding subscriptions provided by an entity would be <tag, value> pairs with wild card operators, SQL queries on the properties, regular expressions that should be evaluated against the verbose text and finally XPath or XQuery queries on the XML document describing the content.

Upon receipt of a message (previously issued by a publisher) the broker checks the topic information contained within the message with the list of previously registered subscriptions. The broker then proceeds to route the message to subscribers with *matching* subscriptions. Here matching refers to the process of evaluating the stored subscriptions against the topic information contained within the message.

The work presented in this paper focuses on the secure end-to-end delivery of messages within publish/subscribe systems. This work addresses the four critical issues related to secure end-to-end delivery of messages viz. confidentiality, authorization, integrity and non-repudiation. The work ensures that the message contents are encrypted for confidentiality, and that the contents can only be deciphered by authorized entities that demonstrate possession of valid credentials. Actions initiated by an entity, such as publishing and subscribing, need to demonstrate appropriate authorizations, the lack thereof results in the failure of the corresponding actions. The framework also allows for enforcing the duration for which the aforementioned authorizations are valid. To enable tamper-evidence (message integrity) and to verify the source (non-repudiation) of a published message, all messages are signed by the publishers.

The framework addresses issues related to revocation of entity credentials, and copes with attack scenarios such as denial of service attacks and replay attacks. We have implemented the framework outlined in this paper and we also report on our benchmarks of this implementation. These benchmarks target several aspects of the security framework, and, we believe, demonstrate the feasibility of our proposed framework.

This research leverages well known cryptography techniques such as encryption and decryption to ensure confidentiality and PKI-based signing and verification of messages. This paper is not about the development of new cryptography algorithms. Similarly, in our section on coping with security attacks we do not consider cryptographic attacks

that target vulnerabilities in the cryptography algorithms that we leverage such as AES and RSA.

This paper is organized as follows. In section II we provide a brief overview of the NaradaBrokering system, and the topic discovery scheme which is leveraged within our security framework. Section III provides a brief overview of the security framework, while section IV describes this framework in detail. In section V we describe how the framework copes with various attack scenarios. We include benchmarks from our implementation of this framework in section VI. Related work is included in section VII, and we outline our conclusions and future work in section VIII.

II. The NaradaBrokering System

NaradaBrokering (http://www.naradabrokering.org) [1-3] is an open-source, distributed messaging infrastructure based on the publish/subscribe paradigm. The subscription formats supported by the messaging infrastructure include Strings, Regular expressions, SQL queries, XPath queries and comma-separated tag=value pairs. For a given topic the system provides services for reliable and ordered delivery. Additional services include Network Time Protocol based synchronized timestamps [2] at distributed entities, buffering services to reduce jitter in multi-media settings, replay and recording services and finally services for broker and topic discovery.

A. The Topic Creation and Discovery Scheme

Interactions between entities in publish/subscribe systems are predicated on the knowledge of the topic that will be used for communications; the publisher will publish over this topic while the subscriber registers a subscription to this topic. The topic discovery and creation scheme [3] in NaradaBrokering facilitates the creation, advertisement and authorized discovery of topics by entities within the system. The discovery process is a distributed process and is resilient to failures that might take place within the system. Topic owners can advertise their topics and can also enforce constraints related to the discovery of these topics. Specifically, a topic owner may require that entities present appropriate credentials (a X.501 security certificate [4]) prior to discovering a topic that it owns. This discovery scheme provides solutions for --

- Provenance –- Verify topic ownership.
- Secure discovery –- A topic owner can restrict the discovery of a topic only to authorized entities or to those that possess valid credentials.

These capabilities are provided by specialized nodes – Topic Discovery Nodes (TDNs) – within the system. Since a given topic advertisement will be stored at multiple TDN nodes, this scheme easily sustains the loss of TDN nodes due to failures or scheduled downtimes.

III. The Security Scheme Overview

This section provides an overview of the objectives and operations involved in the secure delivery of messages. Subsequent sections explain the scheme in greater detail. A message comprises the topic information, message headers and finally the content payload. During secure communications it is the content payload of message that is secured. A topic over which communications need to be secure is referred to as a *secure topic*; this would involve authorized publishing and subscribing, in addition to message payloads being encrypted and signed for confidentiality and integrity/tamper-evidence respectively. Associated with every secure topic is a secret symmetric key that is maintained at a Key Management Center (KMC). There can be more than one KMC within the system and a given KMC can manage more than one secure topics. The KMC is also responsible for generating security tokens that are presented by entities to facilitate authorized publishing and consumption of messages.

The secret key associated with a topic is distributed securely to the interested entities (described in section IV-D). A publisher encrypts the content payload of the message with this secret key. To ensure integrity of the payload, this publisher also signs the encrypted payload; this involves computing the message digest of the encrypted payload and encrypting this hashed value with an asymmetric (e.g. RSA) private personal-key. 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.

Upon receipt of secure messages, an authorized subscriber can validate the signature – to verify the source and to confirm message integrity – and then proceed to decrypt the encrypted payload using the previously distributed secret key.

Based on the security tokens associated with the actions initiated by the publisher and subscriber, brokers that are part of the messaging infrastructure can enforce authorization rules and prevent (or restrict) the dissemination of content.

A. Notations Used in this Paper

A secret symmetric key is represented by K. A public key associated with an entity X is represented as KU_X while the corresponding private key is represented as KR_X. The symmetric key (secret) associated with a Topic T is represented as K_T. The encryption operation E using a symmetric key K over message M resulting in a cipher-text C is represented as $C = E_K(M)$ and the corresponding decryption operation D is represented as $D_K(C) = M$.

The signing operation S, by an entity X, using a hashing algorithm H over a message M is represented as follows $S_X(M) = E_{KR_X}[H(M)]$. The corresponding verification operation V using the signing entity's public key is represented as follows: $V_X(M) = D_{KU_X}[S_X(M)]$; the verification is a success if the result is $H(M)$.

IV. The Security Scheme

We now present details about our security framework. Fig 1 depicts various components of this framework. There is exactly one Certificate Authority within the system. Certificates are assigned to the entities in an out-of-band fashion. The Certificate Authority is not connected to the broker network. There can be one or more KMCs and TDNs

within the system. There can be several clients within the system. All KMCs, TDNs and Clients are connected to one of the brokers within the broker network cloud.

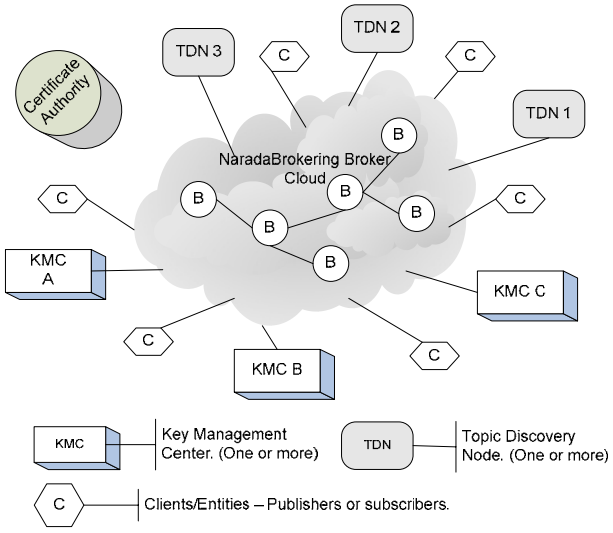


Fig 1: Components involved in the Security Framework

A. The CA within the system

In our scheme we have one Certificate Authority (CA) within the system. The most important function that a CA is responsible for is the issuing of certificates to entities within the system. These certificates can also identify an entity as an authorized KMC or TDN. The CA is also responsible for managing revocation lists pertaining to compromised or rogue entities within the system. The CA also notifies brokers and KMCs within the system about any additions to these revocation lists. A newly added broker or KMC may also request the entire set of revoked certificates from the CA. Please note that the certificate revocations are typically done in an out-of-band fashion.

B. Key Management Center (KMC)

A KMC is a specialized node within the system which is responsible for managing information pertaining to secure topics. There can be more than one KMC within the system, and a given KMC may manage more than one secure topic. However, a given secure topic can be managed by only one KMC. A given KMC performs four core functions. First, the KMC is responsible for the generation of the secret symmetric key that is used for encrypting and decrypting content payloads. Second, the KMC maintains the list of authorized entities associated with a secure topic. In addition to this, the KMC maintains authorization information related to each of these entities. Some of these entities may be registered to publish, subscribe or both.

The third function performed by the KMC is the generation of security tokens. All actions (such as publishing and subscribing) initiated by entities, related to a specific secure topic, require the presentation of the security token. The KMC generates a security token for every authorized entity. This security-token establishes an entity's rights (publish, subscribe or both) over a secure topic and the duration for which these rights are valid. This security token comprises the following:

- Client Certificate, including the identifier or the Distinguished Name.
- Rights – publish, subscribe or both
- Duration for which these rights, and, in fact the token itself, is valid.

The security token A associated with a Topic T for an entity X, is $A_T^X = S_{KMC_T}(C_X, Rights, T, Duration)$. Entities are expected to include this security token along with every action they initiate with individual brokers within the infrastructure. To enable tamper-evidence the contents of this security-token are hashed and signed by the KMC.

Third, the KMC is also responsible for the secure distribution of secret keys and secure tokens associated with a secure topic. To do this the KMC encrypts the contents of the message with a secret key. This secret key is then encrypted using the entity's public personal-key. Only the entity that is in possession of the corresponding private personal-key is able to decrypt the contents of the communications.

Finally, the KMC also sets up a topic over which authorized entities may communicate with it through the exchange of messages. In its topic advertisement to the TDNs, the KMC can also specify restrictions pertaining to the discovery of this topic. Only, entities that are authorized or those that have the right credentials would be authorized to discover the aforementioned topic. No entity will communicate[1] directly with the KMC; no entity, except the KMC administrator and the broker that KMC is connected to will be aware of the physical location of the KMC. This provides an additional degree of insulation from denial of service attacks that are targeted at the host and assorted port numbers over which the KMC listens to for communications.

C. Registering a secure topic

To ensure secure communications over a topic, the topic owner first needs to register the secure topic with a KMC. To do this, the topic owner first needs to discover KMCs that are willing to host the secure topic. When the topic owner initiates this discovery request if it had valid credentials there would one or more responses (containing the advertisements) from the TDN identifying the topics over which the corresponding KMCs communicate. Since a KMC may restrict discovery of KMC-Topic based on the presentation of appropriate credentials, it is possible that a topic owner will not be able to retrieve topic information pertaining to some of the KMCs within the system.

Based on these responses, the topic owner decides the KMC that it would communicate with. The topic owner then registers the secure topic with a KMC while specifying parameters about the symmetric secret key that would be associated with the secure topic. These parameters include the encryption algorithm, the key size and the padding scheme to be used. If there are problems with any of these parameters the

[1]. All communications with the KMC are based on sending messages to the topics that a KMC listens to.

KMC will report problems back to the topic owner. Additionally, the topic owner may specify the Hashing scheme to be used for signing the security tokens generated by the KMC. Having a larger digest increases the integrity of the message; in the case of smaller digests a malicious user can exploit vulnerabilities in collisions arising from the hashing function employed to compute the digest. In our scheme by default we use the 160-bit SHA-1 [5] based message digests.

Trade offs between encryption strength and the performance of the encryption algorithms need to be considered while determining the key lengths for encryptions. Also, shorter key lifetimes, in general, tend to mitigate the effects of lost/stolen keys.

The topic owner then proceeds to register the entities authorized to communicate over the secure-topic, the rights (publish/subscribe) and the duration for which these rights are valid. This information is used to create the security token associated with every authorized entity.

D. Entity KMC communications

All communications between the entities and the KMC need to be secure. To ensure this, all exchanges between the entities and KMC are encrypted using the following rule. First, a secret symmetric key is generated at the sender, and then used to encrypt the content payload. Second, depending on the direction of the communication this secret key is then secured using the KMC's or the entity's public personal-key. Only the entity or the KMC that is in possession of the corresponding private personal-key is able to decrypt the secret key that was used for encrypting the content payload.

This method leverages both symmetric and asymmetric key encryptions. Specifically, asymmetric encryptions have higher overheads for large payloads. By restricting the use of asymmetric encryptions (and subsequent decryptions) to operate on only the secret key, which would typically be a 256-bit AES key [6], we have worked around the high overhead constraint for asymmetric encryptions/decryptions.

E. Distributing keys and security tokens

An entity interested in communications over a secure topic, discovers the topic over which the KMC managing this secure topic communicates. Once this KMC-interaction topic has been discovered the entity issues a request message to the KMC to retrieve the Secret-Key associated with the secure-topic. In this request the entity also includes its credentials and the topic over which any responses should be issued.

Upon receipt of this secret-key request, the KMC first checks to see if the entity is authorized to receive the secret secure-topic key. If the entity is indeed authorized for communications over the secure-topic, the KMC securely routes this secret key to the entity based on the strategy outlined in section IV-D. In addition to the secret key that was routed to the entity, the KMC also routes the corresponding security token (discussed in section IV-B) associated with the requesting entity. This security-token establishes an entity's rights (publish, subscribe or both) and the duration for which these rights are valid.

An entity is expected to include this security token for all actions and exchanges related to this secure topic. These actions include publishing messages or subscribing to the secure-topic. A broker will not perform the expected actions if the entity fails to include this token in exchanges related to this topic (see section IV-H).

F. Publishing messages

When a publisher is ready to publish a message, it performs three steps. First, it encrypts the content payload of the message with the secret secure-topic key that it received from the KMC. Second, the entity also includes its security-token within this message. Finally, the publisher then proceeds to sign the message by computing the message-digest of the encrypted content payload and then encrypting this computed message-digest with its private-personal key. This digital signing enables subscribers to verify the integrity of the message by checking to see if the message has been tampered .

A message comprises a set of message headers and the message payload: $M = M^H + M^P$. We secure both the headers and the body of the message. We do not need confidentiality for the headers, but we do need tamper evidence. In the case of the message payload, we need both confidentiality and tamper-evidence. The message header associated with message is $M^H = S_X(M^P) + A_T^X$. Finally, the message published by a publisher X is $S_X(M^H) + M^H + E_{K_T}(M^P)$. Fig 2 provides an overview of the security related operations at an entity.

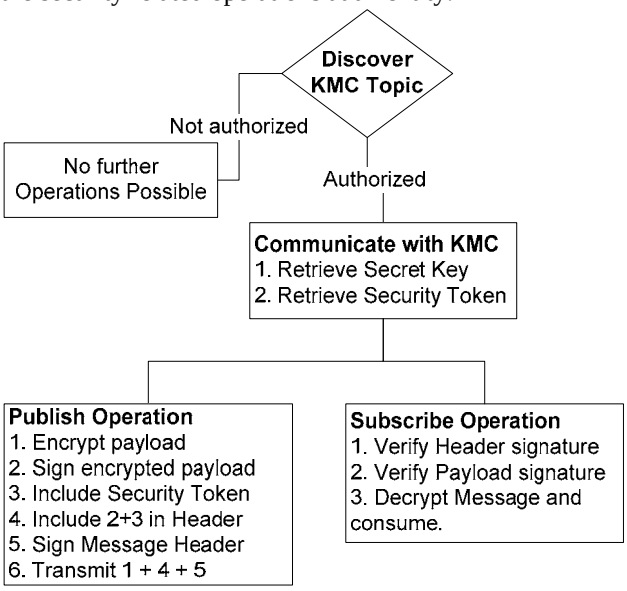


Fig 2: Overview of security operations at an entity

G. Subscribing to secure topics

When an entity is ready to subscribe to a secure topic it includes the security token, assigned to it by the KMC, in its subscription request. Failure to include this security token in its subscription request would prevent the messaging infrastructure from routing messages which match the specified subscription constraint.

H. Broker Operations

All entities within the system use the broker, which it is connected to, to funnel messages into the messaging infrastructure. The broker performs crucial functions (summarized in Fig 3) related to message routing.

1) Keeping track of revoked certificates

In order to verify the credentials and signatures associated with messages and exchanges a broker needs to keep track of revoked certificates. When a broker starts up for the first time, it retrieves the list of revoked certificates (including those issued to KMCs and TDNs). A broker may store (and retrieve) the list of revoked certificates onto (and from) a stable storage. Thus, between successive re-starts a broker only needs to retrieve certificates that have been revoked in the interim.

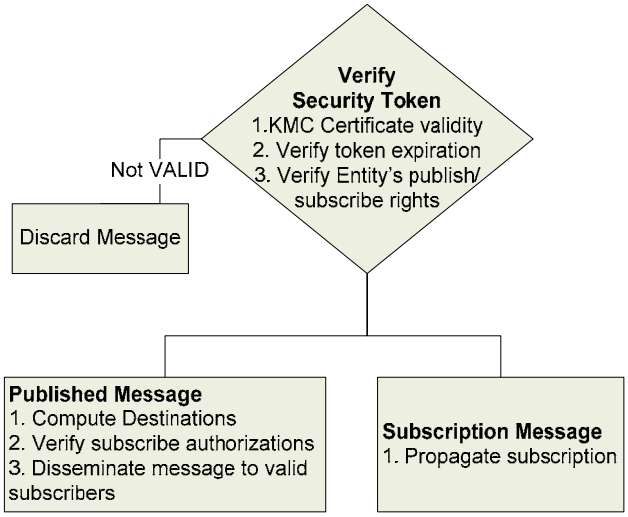


Fig 3: Broker Operations

2) Keeping track of secure topics

Brokers (newly added or otherwise) know about secure-topics as soon as they receive messages targeted to a secure-topic and containing valid security tokens. When a broker joins the broker network this broker retrieves the list of secure-topics maintained by the broker that it connected to.

If a broker receives messages targeted to a secure-topic without a valid security-token it will discard the message without any further processing and concomitant routing. In case a broker is not aware of a secure-topic this broker may propagate the message without a security token (and without performing steps outlined in the next sub-section). A broker that encounters such a message, on what it knows to be a secure-topic, issues an error-message back to the broker that it received the message from. Upon receipt of such a message, the original broker updates its secure-topics list to include this missed secure-topic.

3) Handling messages published by a publisher

Upon receiving a message, issued over a secure topic, from an entity the broker checks to see if there is a security token associated with the message. If the topic is a secure topic and there is no security token associated with the message, the broker discards the message without performing any checks and concomitant routing.

If on the other hand there is a security token associated with the message targeted to a secure topic, the broker performs three functions – verifying if the security token is valid, check topic information contained in token and check for rights associated with the token. If any of these three checks fail the message is simply discarded; if the publisher is connected to the broker and error message is issued.

To verify that the security token is a valid one, the broker first checks to see if the certificate associated with the KMC, which signed this token has been revoked and also if the token has been tampered with, by verifying the signature associated with the security token. Next, the broker checks to see if the topic in the token matches the topic contained within the message. Finally, the broker checks to see if the security token indicates that the publisher that signed the message, indeed has publish permission reflected in its security token. If these checks are completed successfully, the broker proceeds to route the message within the messaging infrastructure.

4) Routing messages to subscribing entities

When a broker receives a subscription request to a secure topic if there is no security token associated with the request, the request is discarded and an error message is issued back to the entity. If there is a security token associated with the subscription request, the broker performs three checks – validity of the security token, topic information and subscription rights. If either of these checks fails the request is discarded and an error message is issued back to the subscribing entity.

A broker will not route messages, corresponding to a subscription to a secure topic, under two distinct conditions. In the first scenario the token associated with the subscription is no longer valid. This might be because either the certificate of the KMC that issued the token was revoked or because the duration of validity within the security token has elapsed.

In the second scenario, it might be possible that a subscription without a security token was allowed by a certain broker that was not aware of a certain secure topic. When the broker encounters a message with a valid security token, it determines the matching subscriptions for the message; if any of these subscriptions do not have a security token associated with them, that subscription is removed from the list of subscriptions maintained at the broker.

V. Coping with various attack scenarios

In this section we outline our strategy to cope with some attack scenarios.

A. 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. In the event that someone tries to

overload the KMC by sending multiple messages to the topic over which the KMC communicates; the KMC will generate a new topic for communications and unsubscribe to messages issued to the old (and compromised) topic. In general it is difficult to "guess" the KMC topic since it based on a randomly generated 128-bit UUID.

No one, except the KMC administrator and the broker that the KMC has connected to, is aware of the physical network address and ports associated with the KMC. It is thus quite difficult to launch a direct denial of service attack on the KMC. In the unlikely event that this information has been compromised, and a denial of service attack was launched based on the network address and port numbers, this particular vulnerability is addressed by rejecting communications from IP addresses that have made multiple bogus attempts.

B. Thwarting replay attacks

It is possible for a rogue entity to capture valid messages and continually publish these messages. To thwart this, a broker keeps track of timestamps associated with the messages published over secure-topics by authorized entities. For every authorized entity the broker maintains the timestamp associated with the last secure message published by that entity. If the timestamp associated with a message is less than or equal to the timestamp associated with the last message published by the entity in question, the message is discarded as a duplicate.

Some schemes resort to maintaining a list of identifiers (typically UUIDs) associated with published messages to determine duplicates. This scheme obviates the need to maintain such identifier information. If the timestamps have millisecond resolution, this limits the rate at which secure messages could be published by an entity to 1000 per second. This limitation is circumvented by also including sequence numbering if the timestamps for two messages are identical. In this case, a broker maintains the timestamp and the sequence numbering within this timestamp. For two messages issued by an entity with identical timestamps a broker will not reject the second message so long as the sequence numbering for the second message is greater than the first one.

This scheme is used in tandem with the global Network Time Protocol (NTP) based timestamps within NaradaBrokering. This enables a broker to discard an old message based on the NTP timestamps within the message.

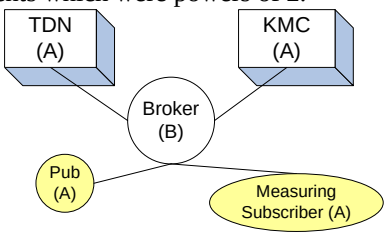
VI. Experimental Results

We have measured several aspects of the security framework, so that the reader has a precise idea of the costs involved in secure communications. All processes executed within JVM 1.4.2, and the cryptography package used was BouncyCastle (http://www. bouncycastle.org) v1.3. In our benchmarks we have used the topologies depicted in Figure 4, the machines hosting the various components are listed in parentheses. In all cases, to obviate the need for clock synchronizations, the publisher and the *measuring* subscriber (which reports the results) were hosted on the same machine. All machines (Table 1) involved in the benchmarks had Linux as the OS, and were hosted on a 100 Mbps LAN.

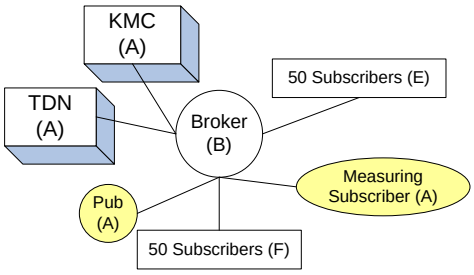
TABLE 1: MACHINES INVOLVED IN THE BENCHMARKS

Machine	Configuration
A	Pentium IV, 2.53GHz, 512MB RAM
B, C, D, E, F, G, H	4 CPU (Xeon, 2.4GHz)machine, 2GB RAM

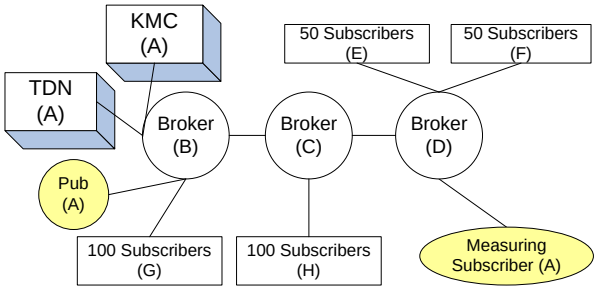
Messages issued by the publisher are time-stamped and upon receipt at the subscriber (after traversal over the network to the broker and then to the subscriber) the transit delay is computed. For the settings in which NaradaBrokering have been deployed – such as the routing of GIS sensor data, A/V multimedia traffic and collaboration messages – the payload sizes do not typically exceed 16 KB. In our experiments, the payloads for messages were varied from 16 bytes to 16KB in increments which were powers of 2.



Topology I: 1 Broker, 1 Subscriber



Topology II: 1 Broker, 101 Subscribers



Topology III: 3 Brokers, 301 Subscribers

Figure 4 Benchmark Topologies

Table 2 summarizes the costs – for a 16KB payload, 256-bit AES key, 1024-RSA PKI and 160-bit SHA-1 – involved in the various operations related to the secure delivery of messages. These costs have the 3 main components –

- Publisher costs: This includes the costs related to computing the initialization vector for a payload, encrypting the message payload with the topic secret key, signing the message and including the security token assigned to it by the KMC.

- Broker costs: This includes the costs for validating the security token, verifying the publisher's signature, computing the destinations and finally routing the message to authorized subscribers.
- Subscriber costs: This includes the costs for validating the publisher signature and decrypting the content payload using the secret topic key that it had previously retrieved from the KMC.

The publisher, broker and subscriber costs were computed separately on machine A. End-to-End costs are based on the topologies depicted in the experimental setups.

TABLE 2: TIMING MEASUREMENTS IN MILLISECONDS. UNLESS OTHERWISE NOTED, THE TABLE REFLECTS COSTS FOR TOPOLOGY-I OVER 100 MBPS LAN FOR A CONTENT-PAYLOAD OF 16KB WITH THE FOLLOWING CRYPTOGRAPHIC PROFILE: ENCRYPTION ALGORITHM {AES 256, PKCS7 PADDING WITH CBC MODE}, SIGNING {1024-BIT RSA, 160-BIT SHA-1}

Operation	Mean	Standard Deviation	Error
End-to-End Delivery (16KB payload)			
1 Broker, 1 Subscriber	64.5	1.32	0.13
1 Broker, 101 Subscribers	83.77	13.93	1.502
3 Brokers, 301 Subscribers	129.6	12.78	1.32
Publisher Costs			
Initialization Vector	1.108	0.025	0.003
Encryption	1.421	0.055	0.005
Signing			
Payload	15.518	0.126	0.013
Header	15.238	0.112	0.011
Broker Costs			
Token and Message Validation	6.989	0.199	0.020
Replay-attack check	0.031	0.005	0.0
Subscription validity check	0.027	0.004	0.0
Subscriber Costs			
Verify Token + Header Signature	3.74	0.13	0.013
Verify Payload Signature	1.64	0.032	0.003
Decryption	1.41	0.021	0.002
KMC: Secure Topic Management			
Generate Secret Key	2.91	0.43	0.04
Generate Signed Security Token	54.89	4.67	0.48
Retrieval: Security Token and Secure Key	71.38	7.22	0.73

Fig 5 depicts the costs involved in the secure end-to-end delivery of messages – at the measuring subscriber for topologies depicted in Figure 4 – for different payload sizes. Costs related to encryption and decryption increase as the payload size increases, this translates to higher overheads for end-to-end delivery. Furthermore, at each broker as the number of targeted subscribers increase the average cost of routing a message to a subscriber increases; this also results in an increase in the time that a message spends in the queue awaiting processing. Finally, as the number of hops increase, broker costs are added to the final transit delay at the measuring subscriber.

The overheads for secure end-to-end delivery are well within the real-time constraints (typically around 100-200 milliseconds) for several interactive collaborative applications such as audio/video conferencing, and distance-education. These overheads are also acceptable for the secure routing of GIS streams by NaradaBrokering in the SERVOGrid project (http://www.servogrid.org/). Successive messages would all be delayed by the amount depicted in our graphs. However, the inter-message spacing for all messages in the stream would be preserved in our scheme. This makes the scheme suitable for applications that can sustain this initial delay.

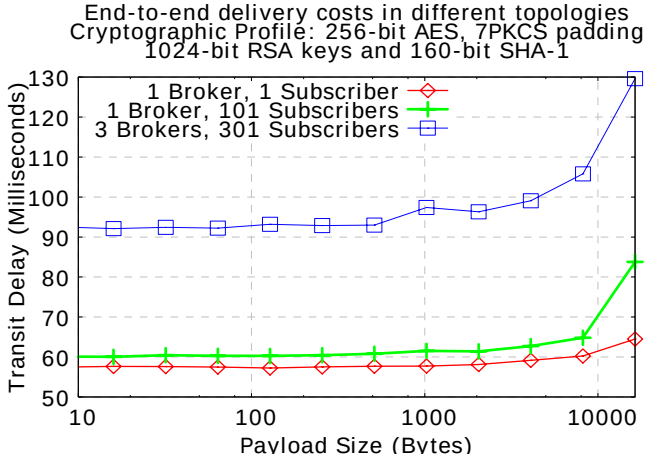


Fig 5: End-to-End costs in different settings

For a given secure message, the latency experienced at a subscriber that is N hops away from the publisher can be approximated as the sum of the publisher costs, the processing at *(N-1)* intermediate brokers, the subscriber costs and the average communication latency for N hops: $t_{pub}^{cost} + (N-1)t_{broker}^{cost} + t_{sub}^{cost} + Nt_{hop}^{cost}$. In wide area network settings, communication costs can be in the order of 20-100 milliseconds, the latency at the subscriber N hops away from the publisher is approximately: $(N-1)t_{broker}^{cost} + Nt_{hop}^{WAN}$.

Presently, we cryptographically verify the security token associated with every message. For a given topic and publisher, this authorization token is typically set to expire after several hours. At the broker, a check is also made to see if the certificate associated with the KMC, which generated the token, is now part of the Certificate Revocation List.

Incorporating optimistic strategies, where a copy of a verified token is stored in memory, and subsequently compared with those contained in the message will accrue significant savings, especially in settings involving a large number of broker hops, since this verification cost is borne at both the broker and the subscriber. The entries in memory would be invalidated after a certain time (a few minutes) has elapsed or a certain number of messages (say 100) have been routed after the last successful verification for the token associated with a publisher/topic pair. We expect this to accrue significant savings, especially in settings involving a large number of broker hops, since this verification cost is

borne at both the broker and the subscriber. However, in this scheme it is possible that, for a small interval, messages would be routed even if the subscriber's duration has elapsed or if the KMC's certificate is revoked. This issue needs to be investigated further.

VII. Related work

GKMP [7] outlines architecture for the management of cryptographic keys for multicast communications. GKMP creates keys 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. Ref [8] outlines 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 schemes to incorporate trust and reputations. Groove [9] provides 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 [10] 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. Ref [11] presents a categorization and details of several attacks scenarios in P2P hash lookup systems, and suggested defences in some of these cases. An excellent survey of security issues that occur in P2P routing protocols, as well as fairness and trust issues that occur in file sharing and other P2P applications is presented in Ref [12].

The Grid Security Infrastructure (GSI) [13] 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 being 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. Legion objects communicate within a secure messaging framework [14] with an abstract authentication/identity system that may use either PKI or Kerberos. Legion also defines an access control policy on objects.

VIII. Conclusions & Future Work

In this paper we presented our scheme for enabling secure end-to-end delivery of messages in publish/subscribe systems. We also presented performance numbers from our implementation of the framework. We believe that these benchmarks demonstrate that the costs introduced by our security scheme are acceptable.

As part of future work, we intend to address security compromises. One of the ways to detect security compromises is to issue authentication challenges at regular intervals along with shorter key lifetimes. In our scheme 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 actions need to be taken to mitigate the affects of the compromise. First, *n*ew keys need to be generated for the secure topics that the entity can publish and subscribe to. Second, *a* message also needs to be propagated throughout the system (to brokers and entities alike) propagating the invalidity of the affected entity's signature and the affected secure-topics.

Acknowledgements

This research is supported by grants from the National Science Foundation's Division of Earth Sciences project number EAR-0446610, and the National Science Foundation's Information and Intelligent Systems Division project number IIS-0536947.

References

[1] S. Pallickara and G. Fox. NaradaBrokering: A Middleware Framework and Architecture for Enabling Durable Peer-to-Peer Grids. Proceedings of the ACM/IFIP/USENIX Middleware Conference. 2003. pp 41-61.

[2] H. Bulut, S. Pallickara and G. Fox. Implementing a NTP-Based Time Service within a Distributed Brokering System. Proceedings of ACM PPoJ. pp 126-134.

[3] S. Pallickara, G. Fox and H. Gadgil. On the Creation & Discovery of Topics in Distributed Publish/Subscribe Systems. Proc. of IEEE/ACM GRID 2005. Seattle, WA.

[4] S. Chokhani and W. Ford Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework, RFC 2527. March 1999

[5] The Secure Hash Algorithm (SHA-1) specified in FIPS 180-1.URL: http://www.itl.nist.gov/fipspubs/fip180-1.htm

[6] J. Daemen and V. Rijmen. AES Proposal: Rijndael, http://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael.pdf

[7] H. Harney and C. Muckenhirn. Group Key Management Protocol (GKMP) Specification. IETF RFC 2093. July 97.

[8] 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.

[9] Groove Networks Inc..http://www.groove.net/

[10] Roger Dingledine, et al. A Reputation System to Increase MIX-net Reliability. Proceedings, Information Hiding Workshop, Mar 2001.

[11] Emil Sit and Robert Morris. Security Considerations for Peer-to-Peer Distributed Hash Tables. Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS. 2002. pp 261-269.

[12] Dan S. Wallach: A Survey of Peer-to-Peer Security Issues. International Symposium on Software Security (ISSS).2002: 42-57

[13] I. Foster, et al. A Security Architecture for Computational Grids," Proc. 5th ACM Conference on Computer and Communications Security Conference, pp. 83-92, 1998

[14] A Ferrari et al. A Flexible Security System for Metacomputing Environments. (HPCN Europe 99), April 1999