CERIAS Tech Report 2006-37

SECOS: KEY MANAGEMENT FOR SCALABLE AND ENERGY EFFICIENT CRYPTO ON SENSORS

by Issa Khalil, Saurabh Bagchi

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

Timely Dissemination of Confidential Events in Content-Based Publish/Subscribe Systems

Yunhua Koglin Elisa Bertino Xukai Zou

Abstract

Content-based publish/subscribe (pub/sub) systems provide more flexibility and expressiveness than subject-based pub/sub systems. They are attractive solutions for distributed event dissemination. In this paper, we address the issue of on-time delivery of confidential events to authorized subscribers, which is the paramount requirement for applications such as the stock trade.

Besides ensured confidentiality, the proposed hierarchical event forwarding scheme among brokers greatly reduces the cost of matching operations performed by a series of brokers; moreover, it enhances event delivery by tolerating broker failures. We also provide an encryption scheme for delivering event from brokers to authorized subscribers. This encryption scheme explores the locality characteristics of subscribers. These characteristics are generated by our hierarchical event forwarding scheme. Our solutions are scalable and particularly suitable for large-scale content-based pub/sub systems. Experimental results validate the high event throughput of our approach.

1 Introduction

Publish/Subscribe (pub/sub) systems provide a new distributed paradigm for event (message) distribution. In these systems, a publisher publishes an event through a broker, also called an event dispatcher. Subscribers specify their interests by registering with a broker. Brokers form a network in which they forward events to each other and, when needed, deliver events to subscribers which have registered with them.

Basically, there are two types of pub/sub systems. The first, referred to as *subject-based* or *type-based* pub/sub, is a system in which events are labeled with predefined subjects to which subscribers may subscribe. The second type, referred to as *content-based pub/sub* system, is more flexible and powerful than the subject-based one. In such a system, both subscriptions and content are specified with respect to a set of *attributes*. An attribute is an ordered pair of *name* and *type*. A subscriber subscribes to events by specifying *predicates* against attributes. For example, if a schema for a stock trade is (*company: string, price: integer, shares: integer*), a subscription could be: $(price < 20) \wedge company =$ *"IBM"*. Because there are no explicit destination addresses associated with an event, brokers are responsible for delivering each event to subscribers whose subscriptions are satisfied by the event, which is called event *matching*. Decoupling publishers from subscribers makes the system scalable and powerful.

In this paper, we focus on the issue in which confidentiality of events needs to be guaranteed and at the same time, events should be delivered on time, because their value decreases with time. Stock trading is one application where such issues are of paramount importance.

Meeting these two requirements can be contradictory, especially in large scale content-based pub/sub systems where the volume of published events is huge. To ensure confidentiality, an event should be encrypted during transmittal, so that only authorized subscribers are able to decrypt it. Usually, a group key shared by both the group members and the bro-

kers is used to encrypt the event. However, since there could be many attributes and thus a large number of complex predicates, for n subscribers, there are possibly 2^n subscription groups that may be interested in an event. Therefore, encrypting the event with group keys could result in a significant performance cost and make the timely dissemination of events difficult.

A simple approach such as multicasting an event by the broker from which the event is published requires replicating subscription information at each broker. However, broker space requirements are a challenge for such approach.

Another consideration for these timely confidential event distribution applications is fault tolerance. A broker failure should not prevent subscribers from receiving events on time. System architecture proposed in [16, 5] where an event is distributed along a spanning tree structure, may involve very expensive reconfigurations [9] if there is a broker failure.

Moreover, the system should minimize *registration information propagating time (RIP time)*, which is the time delay for new subscription information to be propagated into the network. For example, broadcasting an event to each broker, who then distributes the event to authorized subscribers registered from it, has minimal RIP time. Any newly accepted subscribers will get matched events. However, if the broker from which an event is published multicasts directly to authorized subscribers, then it takes time for new subscription to be propagated to each broker, especially when the network is large. Therefore, some newly accepted subscribers may miss some events.

Our contribution We propose an event forwarding scheme called hierarchical event routing. This scheme increases system availability by tolerating some broker failures. Additionally, our approach can efficiently determine the subscription groups to which an event has to be delivered by exploiting locality. We also propose an efficient encryption scheme, under which a broker encrypts an event only once. The encryption key can be efficiently derived by subscribers, even though they may belong to different subscription groups. In our solution, a broker needs only to keep the subscription information of its group, which is a small number of brokers. Therefore, not only are storage requirements reduced for each broker, but also time and network traffic are reduced for subscription information propagation. We provide theoretical proofs that our approach ensures event confidentiality. Experiment results validate the high event throughput of our approach.

The rest of the paper is organized as follows. We describe our application model in Section 2, and then present our hierarchy event routing scheme in Section 3 and our event distribution scheme in Section 4. Experimental results and related work are presented in Sections 5 and 6, respectively. Section 7 concludes the paper and outlines future work.

2 Model

In this paper, we focus on how to ensure that confidential events are delivered on time to authorized subscribers. It is important to note that pub/sub systems involve several security and service quality issues. Here we only address part of them. Other issues, such as event access control policies, integrity and authentication, are not the focus of this paper. Therefore, we assume that brokers are trusted. They will enforce event access control policies when subscribers subscribe from them; and they only accept events published by authorized publishers, and guarantee integrity of events they route.

In our system, a broker may fail, or come under DoS attacks, therefore it may not be available to deliver events. An unauthorized subscriber of an event (whose subscription is denied by brokers, or if accepted, the event does not match this subscriber's subscription) may want to access the event.

Note that even though our approach eliminates the matching performed by brokers while an event is forwarded among them, a matching algorithm is needed when a broker has to decide to which groups of subscribers an event should be delivered. However, such a matching algorithm is likewise not the focus of this paper. We assume that such an algorithm exists and that locality is used in the algorithm for efficiency.

In the next two sections, we describe our schemes for event distribution. It includes two steps: the first one is that an event is routed from the broker from

which the event is published to some brokers, and the second one is the event is forwarded from these brokers to authorized subscribers.

3 Hierarchial Event Routing Scheme

In this section, we describe our event routing scheme among brokers, followed by a discussion of the main features of this scheme.

3.1 Hierarchy Event Routing

In our pub/sub system, all brokers are labeled with an ID, such labeling can be performed by the party responsible for accounting the services of the system.

Definition 1 *A leaf broker group (LBG) with label i is denoted as LBG_i where $LBG_i = \{b_{i1}, b_{i2}, \ldots, b_{im}\}$ such that*

- *brokers $b_{i1}, \ldots, b_{im} \in LBG_i$ are located closely in network topology;*
- *the ID of each broker in LBG has the same prefix as its group ID;*
- *the size of LBG_i is $|LBG_i|$ and $|LBG_i| \geq t$ where t is equal to $\lceil \frac{1}{1-r} \rceil$ and r is the estimated broker failure rate; and*
- *if a subscription request is submitted to and then approved by broker $b_{ij} \in LBG_i$, this subscription will be securely multicasted by b_{ij} to all other brokers in LBG_i.*

Therefore, any broker in a LBG maintains the subscription information of subscribers who are registered with any broker in the LBG.

Based on the labels of LBGs, a tree is formed where LBGs are the leaves and all inner nodes are formed by virtual broker group (VBG). Specifically, a VBG_i has label i and is virtually formed by either all LBG_i* or all VBG_i*.

Example 1 *Figure 1 shows an example of 3-ary tree with a height of 2. The leaves are LBGs. LBG_{11} includes brokers $b_{111}, b_{112}, b_{113}$ and b_{1113}. All these brokers share the same group label prefix (11). Figure 2 illustrates the locality of brokers in Figure 1.*

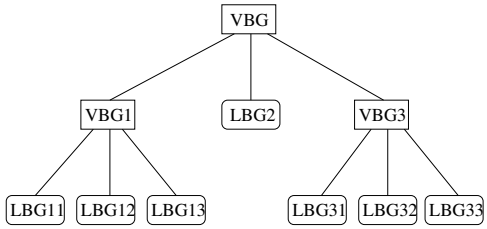

Figure 1: A 3-ary tree formed by broker groups

Brokers in an LBG periodically authenticate each other and exchange their subscription information; however, a broker does not propagate its subscription information to another LBG.

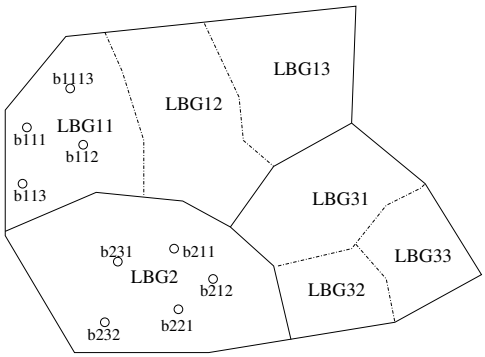

Figure 2: The locality of brokers

An additional information kept by each broker is a forwarding table. If a broker belongs to an LBG at depth h' of the tree, then the table is of dimension $h' \times (n-1)$. Each entry of such a table stores IP addresses of t brokers of other LBGs. Thus, if $r = 10\%$, each entry keeps information about 2 brokers. In such a table, each column has $n-1$ entries. For each of these entries, the label of the stored brokers shares the same prefix as the entry label. These brokers are randomly choosen from their group. Table 1 shows broker 111's table and Table 2 shows broker 213's table.

Table 1: Event forwarding table of broker 111

2*(211, 221)	12* (122, 121)
3*(311, 322)	13* (133, 135)

Table 2: Event forwarding table of broker 213

1**(114, 132)
3**(312, 321)

A broker periodically authenticates those brokers kept in its forwarding table and updates the information in case some brokers are under DoS attack or system failure.

Algorithm 1 Hierarchy event forwarding algorithm

Input: (S, H, T) //S: ID of event sender, H: ID of host, T host's routing table of dimension $h' \times (n-1)$

```
1:  if S = H then
2:    for i = 1 to h' do
3:      for j = 1 to n − 1 do
4:        choose a broker in entry (i, j) and forwards
          the event;
5:      end for
6:    end for
7:  else
8:    let p = max number of matched prefix of S and
      H
9:    for i = p + 1 to h' do
10:     for j = 1 to n − 1 do
11:       choose a broker in entry (i, j) and forwards
          the event;
12:     end for
13:   end for
14: end if
```

Algorithm 1 shows how a broker routes an event. Line 1-5 is the case where the broker is the one where the event is published. In this case, it routes the event to one broker in each of its entry in the routing table. Line 8-13 is the case where the broker receives the event from another broker. In this case, the broker uses the max common ID prefix with the sender to determine to which level it should start forwarding an event (Line 8), and then starts to forward the event to all entries from that level.

3.2 Discussion

We now discuss our routing scheme with respect to several metrics.

Broker Space Requirements: In most previous approaches, a broker must keep subscription information about the whole network. In large scale pub/sub systems, such a requirement implies that all subscription information is replicated at each broker. Under our approach, a broker only needs to maintain $1/n^h$ of the whole network's subscription information, where n and h are the degree and height of the tree, respectively. In a 4-ary tree with a height of 3, this is only 1/64 of the total network's subscribe information.

Subscription Information Update: In most previous approaches, new subscription may need to be propagated to the whole network. Our approach needs only to multicast such information to $1/n^h$ of the network, which greatly reduces network traffic.

Subscription Information Propagation Delay Time: Our approach decreases the delay time by propagating this information only within $1/n^h$ of the network.

Execution of Sequential Matching: Our approach requires only a single broker in a group to perform event matching, thus avoiding the execution of sequential matching which must be performed by brokers in most tree-based approaches.

Failure Tolerance: Our approach achieves the same level of fault tolerance as approaches based on event flooding; however, event delivery is faster in our approach since it takes $O(h)$ for an event to reach a leaf broker. By contrast, under the flooding approach it takes $O(n^h)$ for an event to reach a leaf broker.

Load Balancing: Most tree-based event delivery systems suffer from unbalanced loads. Leaf brokers in the tree seldom perform event matching and forwarding to other brokers, while brokers which are *centroids*[1] of the tree suffer from a heavy load. In our approach, the load is almost uniformly distributed

[1]A centroid of an n-node tree T is a node such that its removal from T leaves no connected components of size greater that $n/2$.

among brokers, if a publisher publishes an event randomly at any broker. Our broker routing table ensures this property.

Matching Cost: Instead of letting one broker perform matching and event delivery to all subscribers in the system, in our approach, event matching and distribution are executed in parallel by n^h brokers, each supporting $1/n^h$ of the load. Also, our scheme possesses locality characteristics and therefore could use caching or popular group matching algorithms for efficient matching.

Dynamic: When a LBG i has been added too many brokers, such a group could be handled as a VBG which is formed by several LBGs and all brokers in previous i are divided into these LBGs. Or when a LBG contains too few brokers as some brokers have been deleted, such a group could be merged with another LBG.

After a broker finishes routing an event, it needs to distribute the event to authorized subscribers, if any exist. Next, we describe how a broker encrypts an event and distributes it to all the groups of authorized subscribers within its broker group.

4 Confidentiality-Preserving Event Delivery

We assume that brokers accept subscription requests by following the policy of the system, such as requiring payment evidence. After a subscriber submits its subscription, if the request is permitted, the subscriber receives one or more keys corresponding to the groups in which its subscription falls. Since our focus is not on event space partitioning algorithms, we denote the event space as follows.

Let $\mathcal{G} = \{G_1, \ldots, G_n\}$ represent all event space of the subscription information in a broker group $\mathcal{B}$.

Group G_i is defined as (S_i, K_i, V_i) where S_i is part of the subscription space, K_i is the secret key shared by all brokers in $\mathcal{B}$ and these subscribers whose subscription predicates p are satisfied by S_i, and V_i is a linked list of subscribers which belong to group G_i. Furthermore, K_i is in $\{0,1\}^l$ where l is a security parameter. For any $G_i, G_j \in \mathcal{G}$, $S_i \neq S_j$. However, we do not require that $S_i \cap S_j = \emptyset$

Note that because brokers in a group share subscription information, they will assign subscribers the same group and the corresponding key. In another word, all brokers in a group keep the same G_i.

Given an event e, a broker must first run a matching algorithm *match (e, $\mathcal{G}$)* which returns $\mathbf{G} \subseteq \mathcal{G}$, that is, a set of groups to which the event should be delivered. Next, the broker encrypts the event so that it can be decrypted only by the subscribers in these groups belong to $\mathbf{G}$.

Because an event may match several groups, encrypting the event several times with different group keys makes event delivery very inefficient. Here we propose an efficient and practical encryption scheme, which has the property that the encryption key is independent from the group keys. However, all authorized subscribers can derive the encryption key. Our strategy is the following:

Assume $\mathbf{G}= \{G_1, \ldots, G_m\}$ are the groups to which the event should be delivered. To encrypt an event, the broker

1. generates a random symmetric encryption key T such that T is in $\{0,1\}^l$ and a nonce r where $r \in \{0,1\}^l$;

2. encrypts the event with T;

3. calculates $A = ([G_1, D_1], \ldots, [G_m, D_m])$ such that $D_i = h(K_i \oplus r) \oplus T$ (for $i = 1, \cdots, m$), where $\oplus$ is XOR operation and h is a secure one-way function $h : \{0,1\}^l \rightarrow \{0,1\}^l$.

Then the broker multicasts the encrypted event to all subscribers in $\mathbf{G}$. In the encrypted event, the broker appends A and the nonce r. A member of group G_i can obtain T by $D_i \oplus h(K_i \oplus r)$ and thus decrypt the event.

4.1 Discussion

In this subsection, we discuss several important correctness properties of our approach. We first provide our correctness criteria and then prove that our encryption scheme satisfies them.

Definition 2 Completeness *Each authorized subscriber should be able to derive the key to decrypt an event.*

Definition 3 Soundness *If an individual is not authorized to access the event, then it is not able to decrypt the event. Further, a member* subj *of some group $G_i \in \mathbf{G}$, knowing T, r and A, is not able to derive any secret key K_j of group $G_j \in \mathbf{G}$ $(j \neq i)$, unless* subj *belongs to group G_j.*

Definition 4 Collusion-resilience *It is impossible for any set of adversaries to derive a secret key that none of them possesses.*

Theorem 1 *The proposed scheme is complete.*

Proof: For an event, an authorized subscriber belongs to at least one of $G_i \in \mathbf{G}$ to which the event falls; thus, it has at least one key K_i. As a result, the subscriber can decrypt T as $D_i \oplus h(K_i \oplus r)$ from the public D_i and r, and the event too. □

Theorem 2 *The proposed scheme is sound.*

Proof: To decrypt an event, an individual must obtain the encryption key T. Since T is mixed in D_is, in order to obtain T, the individual must get at least one $h(K_i \oplus r)$. If the individual is an outsider or a system subscriber but does not belong to any of $G_i \in \mathbf{G}$, the individual cannot compute any of $h(K_i \oplus r)$s. Thus, the individual cannot obtain T.

Consider a subscriber belonging to $G_i \in \mathbf{G}$. Given the nonce r, A and T, the subscriber can obtain T by computing $H_i = h(K_i \oplus r)$ and then $D_i \oplus H_i$. Once knowing T, the subscriber can compute $H_j = D_j \oplus T$ (for any $j = 1, i-1, \ldots, i+1, m$). However, due to the one-way property of h function, the subscriber cannot get K_j from $H_j (i.e., h(K_j \oplus r))$. Thus, the proposed scheme is sound. □

Theorem 3 *The proposed scheme is collusion-resilient.*

Proof: We consider the collusion of subscribers not outsiders since involvement of outsiders cannot contribute any bit of information useful to reconstruct the keys. There are three possible collusion: by the subscribers not belonging to any $G_i \in \mathbf{G}$; by the subscribers belonging to some $G_i \in \mathbf{G}$; and by subscribers across several G_i's in $\mathbf{G}$. For the first case, these subscribers are equivalent to outsiders, and cannot get T or any K_i. For the second case, these subscribers can get their T and any of other $H_j = h(K_j \oplus r)$s. Because the security of the one-way function is independent of the number of users trying to break it, the collusion of these subscribers is no stronger than an individual subscriber trying to get K_j from H_j. The same principle applies to collusion of multiple subscribers from different groups.

As a result, the proposed scheme is resilient to any kinds of collusion attacks. □

4.2 Dynamics and rekeying

In a dynamic pub/sub system, the user can subscribe to or unsubscribe/be revoked from the system, even move from one group to the other. In order to guarantee that all (and only) current subscribers in a group G_i can obtain the events destined to the group, the group key K_i needs to be updated. The updated key needs to be distributed to the users securely. We propose using the same scheme a second time to address such a requirement. During user registration, the system will assign each user a unique personal secret ID (denoted as UID_i for user U_i). As a result, if a new K_i needs to be securely distributed to the current users $\{U_1, U_2, \cdots, U_n\}$ of a group G_i, the following message will be multicast (note: r is a new random nonce): $\{r; h(UID_1 \oplus r) \oplus K_i, h(UID_2 \oplus r) \oplus K_i, \cdots, h(UID_n \oplus r) \oplus K_i\}$.

As for the performance, the proposed scheme is very efficient. The computation of h depends totally on the algorithm selected and is independent of m, so its computation complexity can be considered as constant, such as $O(c)$ where c is a constant. The XOR operation is fast and can be considered to run in $O(1)$ time. Thus, time complexity of computing A is $O(mc)$ (i.e., $O(m)$).

As it can be seen from the discussion, the proposed scheme is secure, efficient, and dynamic and is able to enforce content confidentiality for pub/sub systems.

5 Simulation Results

In this section, we evaluate the efficiency of our approach by simulation.

5.1 Methodology

The simulations were conducted using a 5-ary tree of 3 levels, i.e., there are total 125 broker groups, with each group having 5 brokers. The total number of brokers in the system is 625.

Each broker has 50 subscribers. For simplicity, the event space is partitioned into units. Each subscriber subscribes to 1 unit from the event space.

We experimented with different distributions for event popularity and compared ours with a basic tree-based approach and a multicast-based approach:

- Uniform: Subscribers from any broker subscribe to event space units randomly.
- Normal distribution: Subscribers from a broker i subscribe to event units following a normal distribution. For an event space of n units, the subscription from broker i has a mean at the unit $n*(i+0.5)/625$. We evaluate the effect of different standard deviations: 1/6 of the event space, 5 and 50. This follows from the fact that a subscription has locality characteristics: subscribers within a broker have similar distributions of interests, whereas the greater the distance between brokers, the more different the subscription distributions.

5.2 Results

We present our results on space requirements for a broker, and on the delay in receiving an event by subscribers.

Space Requirements

We measure the space requirement for storing subscription information by a broker. IP addresses are 4 bytes long and event space is 2 bytes. For each event unit, if there is at least one subscription, then it forms a subscription group. For each group, a linked list is used to store subscribers information. Figure 3 shows the result of subscriptions which are uniformly distributed. It reveals that a multicast approach requires a dramatically large space. When comparing our approach with the tree-based approach in detail, our approach uses a relatively larger space than a tree-based approach (see Fig. 4). We think this is an acceptable space requirement (less than 1.4 KB). The results for normal distributions are reported in Fig. 5, 6, 7, 8, 9, 10. They follow similar patterns.

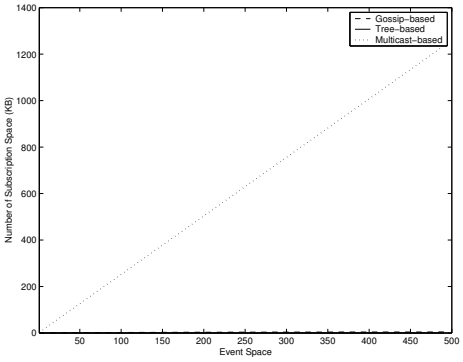

Figure 3: Space requirement for a broker (uniform)

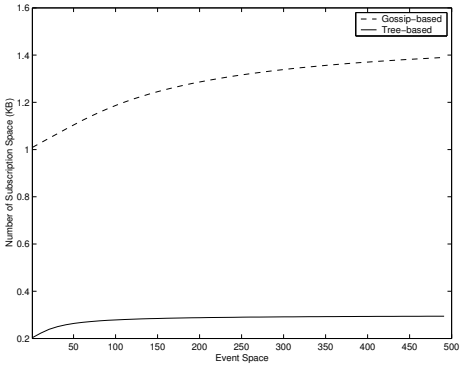

Figure 4: Space requirement for a broker (uniform)

Next, we check the number of subscription groups formed under different subscription distributions. These results are reported in Figure 11, 12, 13, 14, 15, 16. In multicast approach, the number of groups is the same as the number of event space units. When the standard deviation decreases, our approach has a similar number of groups as the tree-based ap-

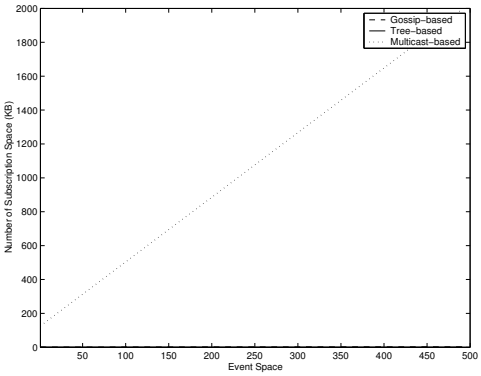

Figure 5: Space requirement ($\sigma = 1/6$ of event space)

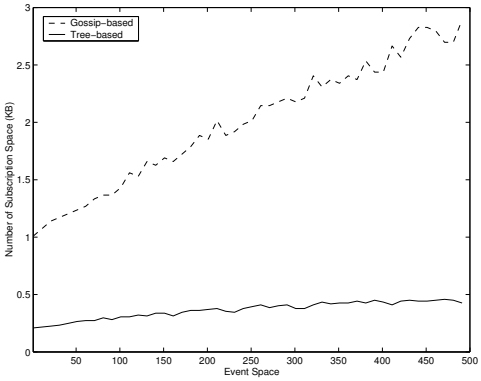

Figure 6: Space requirement ($\sigma = 1/6$ of event space)

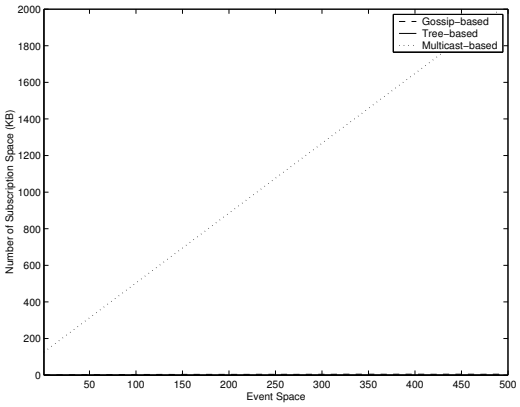

Figure 7: Space requirement for a broker (σ =50)

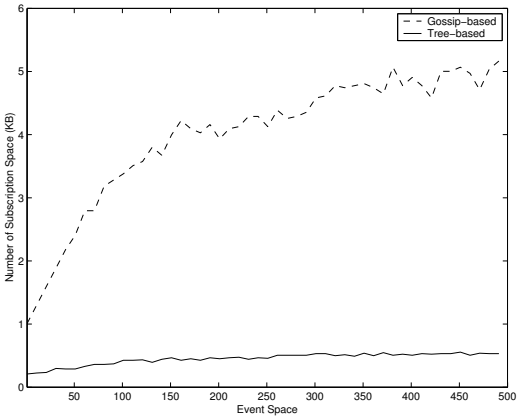

Figure 8: Space requirement for a broker (σ =50)

proach. For a tree-based approach, the routing information is aggregated, therefore, a broker only needs to maintain the subscription information from its subscribers. Even for normal distribution with large deviation, we found that a broker can aggregate these brokers' subscription information into a very small space.

Time Delay

We evaluate the time delay for a subscriber to receive an event. In the evaluation, it takes 1 unit of time to forward an event to a group. Figure 17 reports the time to deliver an event in a uniform distribution. The tree based approach takes more time due to the series of event forwarding by brokers in such a structure. By contrast, multicasting needs to forward to all subscription groups of the whole network. Our hierarchy approach, however, uses parallel event forwarding, which is very efficient. Figure 17 also shows that when the number of subscription groups increases, the performance of a multicast approach degrades; it is slower than the tree-based approach when the event space size reaches 350. Figures 18, 19, 20 report the results for normal distributions. The tree-based approach is relatively stable regardless of the standard deviation. When the standard deviation increases, our approach takes a longer time.

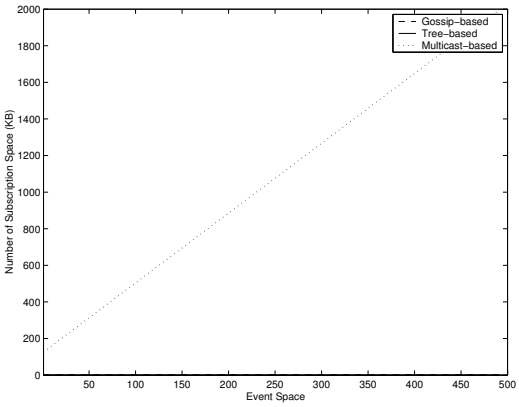

Figure 9: Space requirement ($\sigma = 5$)

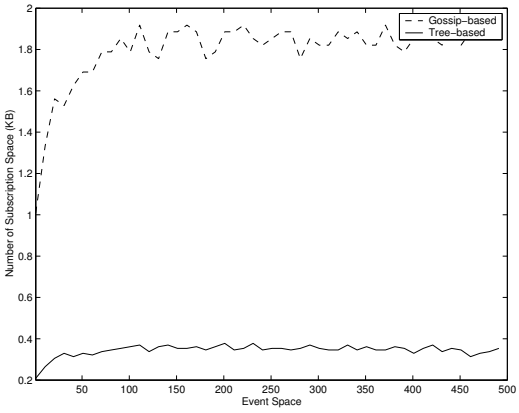

Figure 10: Space requirement (σ =5)

Broker Involvement

The number of brokers involved in forwarding an event reflects the load that a broker takes. Figure 21 shows the number of brokers that are involved in delivery of an event. The subscription rate of events is uniformly distributed. The horizontal axis shows the subscription rate of each *broker*. The vertical axis shows the number of brokers involved in delivering an event. For the tree-based approach, a spanning tree is built by applying Dijkstra's algorithm on a graph of 625 nodes, each node with degree d where d is uniformly distributed from 1 to 4. The result is the average of ten runs, where at each run, an event is published at a randomly chosen broker. We also present the ideal approach, where only the brokers subscribing to an event are those which participate in the delivery of the event. From Fig. 21, the tree-based approach needs more brokers to participate in the event delivery, especially when subscription rate is 20%, half of the brokers involved are not interested in an event. As the subscription rate increases, the tree-based approach reaches the same number as the ideal approach at 100%. In that instance, all brokers in a tree-based approach are interested in the event.

Hierarchical event forwarding uses an almost constant number of brokers for forwarding an event. If there is no broker failure, this number is equal to the number of broker groups and it is not related to the subscription rate. Therefore, when the subscription rate is below 20%, our approach needs more brokers than the ideal approach. Our approach uses less brokers than the tree-based approach when the subscription rate is above 6%. For a broker with 50 subscriptions, this means an event has popularity below 0.12%. Thus, our approach seldom uses more brokers than the tree-based approach.

6 Related Work

Several studies have been devoted to investigating efficiency issues concerning pub/sub systems [11, 10, 2, 1, 13, 2, 3, 6, 15, 8, 9, 5, 12, 17, 7, 14, 4] and several prototype systems have been developed. Most approaches like [8, 9, 5] use a spanning tree structure for event routing. In order to reduce the matching that has to be performed by brokers from the root to the leaves, several optimization techniques have been proposed. Virtual groups are used to reduce the matching performed by brokers [16].

However, security issues [15] in content-based pub/sub systems have not been so widely investigated. Srivatsa and Liu [14] propose a resilient network which, instead of providing only a single path from each publisher to its subscribers which is inherited from the spanning tree structure, several independent paths from a publisher to each of its subscribers are provided. Such paths are built de-

terministically. In their approach, building several independent paths from a publisher to every subscriber involves complex topology computations. In dynamic environments, such computation is expensive. Such expensive reconfigurations of tree structures have been completely eliminated in our hierarchy event forwarding scheme. Each broker maintains a forwarding table which ensures that at least one broker in the next forwarding level is operative.

To avoid unnecessary event broadcasting, Carzaniga et al. [5] proposes an approach that broadcasts events only along the spanning tree. As previously mentioned, in dynamic environments, a tree structure is hard to maintain and may become disconnected. Broadcasting along a disconnected tree involves more redundancy and does not solve the availability problem.

Opyrchal and Prakash [11] discuss how a broker can encrypt an event and deliver it to a possibly very large number of groups. As each group has a secret key shared by members and brokers, encrypting the event using a group key may involve performing many encryption operations, and there may be several groups to which this event should be delivered. Caching and clustering are therefore used to make fewer encryptions. In our confidentiality-preserving encryption scheme, an event is encrypted only once. All authorized subscribers can derive the key for decryption the event efficiently. Our scheme is provably secure.

7 Conclusions and Future work

In this paper, we address some security issues of content-based pub/sub systems. We focus on increasing the availability of events and ensuring confidentiality when events are delivered to authorized subscribers. Our schemes (hierarchical event forwarding and confidentiality-preserving encryption) are efficient and scalable. Our approach is especially suitable for large-scale content-based pub/sub systems. Simulation results validate the efficiency of our approach.

We plan to investigate other security issues in content-based pub/sub systems, especially how brokers can efficiently authenticate each other, and how to ensure confidentiality of events forwarded among brokers. We would like to investigate an efficient mechanism for matching an encrypted event against subscriptions, which is suitable for large-scale content-based pub/sub systems.

References

[1] M. Aguilera, R. Strom, D. Sturman, M. Astley, and T. Chandra, *Matching events in a content-based subscripton system*, Proc. of the 18th ACM Symposium on Principles of Distributed Computing, 1999.

[2] G. Banavar, T. Chandra, B. Mukherjee, and J. Nagarajarao, *An efficient multicast protocol for content-based publish subscribe systems*, Proc. of the 19th IEEE International Conference on Distributed Computing Systems, 1999.

[3] Fengyun Cao and Jaswinder Pal Singh, *Efficient event routing in content-based publish-subscribe service networks*, Proc. of IEEE INFOCOM 2004, 2004.

[4] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, *Design and evaluation of a wide-area event notification service*, ACM Trans. on Computer Systems **19** (2001), no. 3, 332–383.

[5] A. Carzaniga, M. J. Rutherford, and A. L. Wolf, *A routing scheme for content-based networking*, IEEE INFOCOM, 2004.

[6] Antonio Carzaniga and Alexander L. Wolf, *Forwarding in a content-based network*, Proceedings of ACM SIGCOMM 2003 (Karlsruhe, Germany), August 2003, pp. 163–174.

[7] P. Costa, M. Migliavacca, G. P. Picco, and G. Cugola, *Epidemic algorithms for reliable content-based publish-subscribe: an evaluation*, Proc. of the 24th IEEE International Conference on Distributed Computing Systems, 2004.

[8] P. Costa and G. P. Picco, *Semi-probabilistic content-based publish-subscribe*, Proc. of the

25th IEEE International Conference on Distributed Computing Systems, 2005.

[9] G. Cugola, D. Frey, A. L. Murphy, and G. P. Picco, *Minimizing the reconfiguration overhead in content-based publish-subscribe*, Proceedings of the 19th ACM Symposium on Applied Computing, 2004.

[10] A. K. Datta, M. Gradinariu, M. Raynal, and G. Simon, *Anonymous publish/subscribe in p2p networks*, Proc of the Int. Parallel and Distributed Processing Symposium, 2003.

[11] L. Opyrchal and A. Prakash, *Secure distribution of events in content-based publish subscribe systems*, Proc. of the 10th USENIX security symposium, 2001.

[12] G. P. Picco, G. Cugola, and A. L. Murphy, *Efficient content-based event dispatching in presence of topological reconfigurations*, Proceedings of the 23rd International Conference on Distributed Computing Systems, 2003.

[13] Anton Riabov, Zhen Liu, Joel L. Wolf, Philip S. Yu, and Li Zhang, *Clustering algorithms for content-based publication-subscription systems*, Proceedings of the 22nd IEEE International Conference on Distributed Computing Systems, 2002.

[14] M. Srivatsa and L. Liu, *Securing publish-subscribe overlay services with eventguard*, Proceedings 12th ACM Conference on Computer and Communication Security, 2005.

[15] Chenxi Wang, Antonio Carzaniga, David Evans, and Alexander L. Wolf, *Security issues and requirements for internet-scale publish-subscribe systems*, Hawaii International Conference on System Sciences, 2002.

[16] Rongmei Zhang and Y. Charlie Hu, *Hyper: A hybrid approach to efficient content-based publish/subscribe*, In Proceedings of International Conference on Distributed Computing Systems, 2005.

[17] H. Zhou and S. Singh, *Content based multicast (cbm) in ad hoc networks*, MobiHoc, 2000, pp. 51–60.

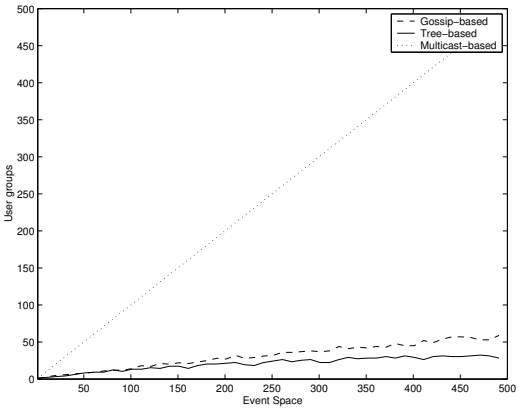

Figure 11: Number of subscription groups ($\sigma = 1/6$ of event space)

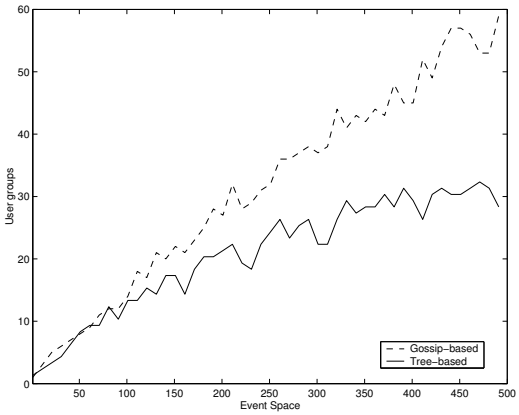

Figure 12: Number of subscription groups ($\sigma = 1/6$ of event space)

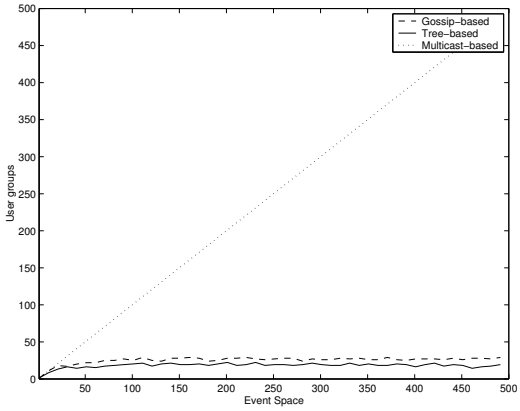

Figure 13: Number of subscription groups ($\sigma = 5$)

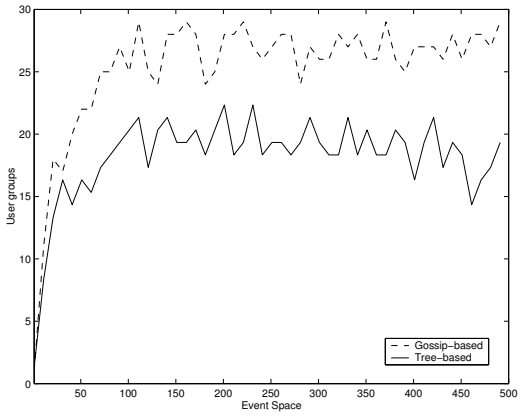

Figure 14: Number of subscription groups ($\sigma = 5$)

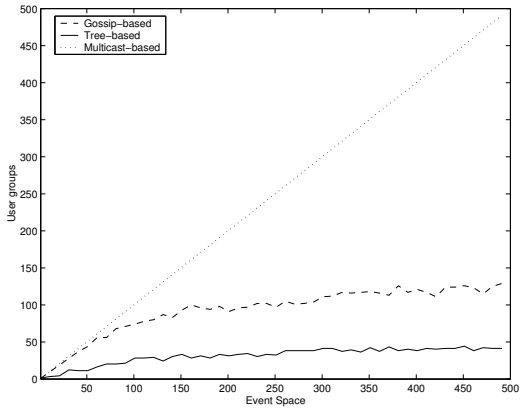

Figure 15: Number of subscription groups (σ =50)

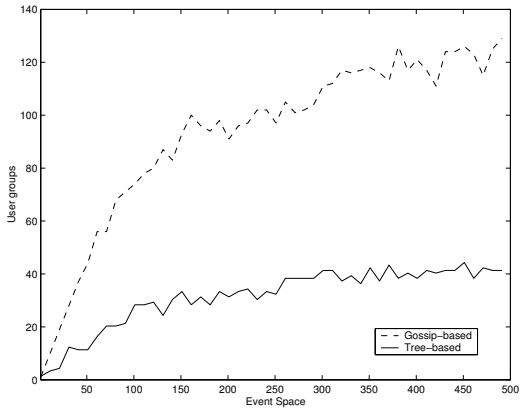

Figure 16: Number of subscription groups (σ =50)

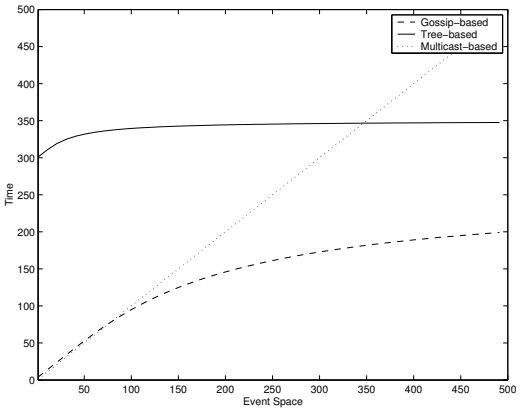

Figure 17: Time delay (uniform)

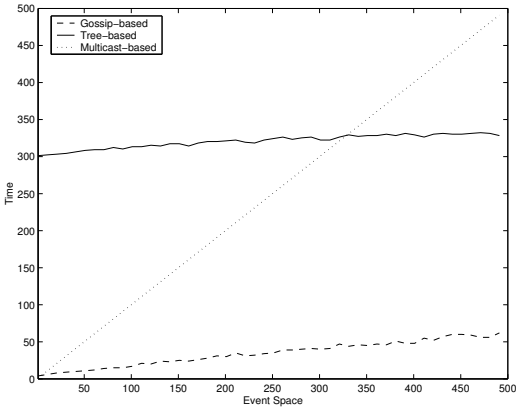

Figure 18: Time delay ($\sigma = 1/6$ of event space)

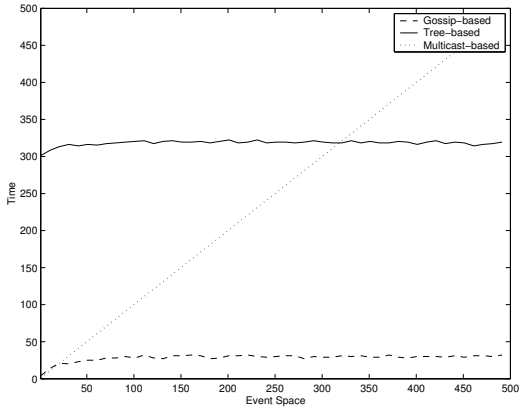

Figure 19: Time delay (σ =5)

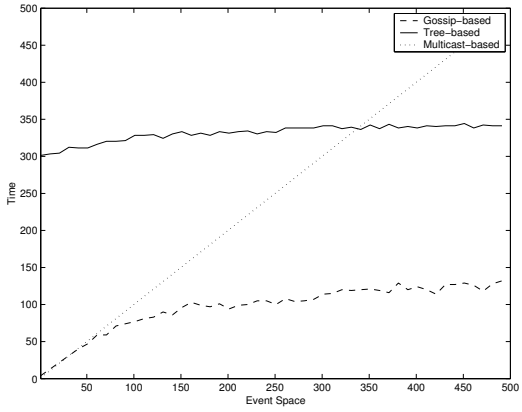

Figure 20: Time delay (σ =50)

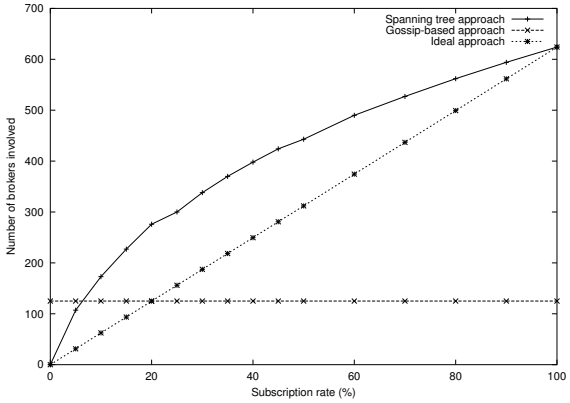

Figure 21: The number of brokers involved