Supporting Publication and Subscription Confidentiality in Pub/Sub Networks*

Mihaela Ion[1], Giovanni Russello[1], and Bruno Crispo[2]

[1] CREATE-NET International Research Center,
via alla Cascata 56D, 38123 Trento, Italy
{mihaela.ion,giovanni.russello}@create-net.org
[2] Department of Information Engineering and Computer Science,
University of Trento, Trento, Italy
crispo@disi.unitn.it

Abstract. The publish/subscribe model offers a loosely-coupled communication paradigm where applications interact indirectly and asynchronously. Publisher applications generate events that are sent to interested applications through a network of brokers. Subscriber applications express their interest by specifying filters that brokers can use for routing the events. Supporting confidentiality of messages being exchanged is still challenging. First of all, it is desirable that any scheme used for protecting the confidentiality of both the events and filters should not require the publishers and subscribers to share secret keys. In fact, such a restriction is against the loose-coupling of the model. Moreover, such a scheme should not restrict the expressiveness of filters and should allow the broker to perform event filtering to route the events to the interested parties. Existing solutions do not fully address these issues. In this paper, we provide a novel scheme that supports (i) confidentiality for events and filters; (ii) filters can express very complex constraints on events even if brokers are not able to access any information on both events and filters; (iii) and finally it does not require publishers and subscribers to share keys.

1 Introduction

The publish/subscribe (pub/sub) model is an asynchronous communication paradigm where senders, known as *publishers*, and receivers, known as *subscribers*, exchange messages in a loosely coupled manner, i.e. without establishing direct contact. The messages that publishers generate are called *events*. Publishers do not send events directly to subscribers, instead a network of interconnected brokers is responsible for delivering the events to the interested subscribers. In fact, publishers do not know who receives their events and subscribers are not aware

* An earlier version of this paper appeared in the Proceedings of the 6th International ICST Conference on Security and Privacy in Communication Networks (SecureComm 2010).

of the source of information. In order to receive events, subscribers need to register their interest with a broker through a *filter*. When a new event is published, brokers forward it to all subscribers which expressed a filter that matches the event.

The pub/sub communication paradigm has the advantage of allowing the full decoupling of the communicating entities [8] which enables dynamic and flexible information exchange between a large number of entities. The communicating parties do not need to know each other or establish contacts in order to exchange content. Moreover, if durable subscription is enabled, publisher and subscribers do not need to actively participate in the interaction at the same time. If a subscriber is offline when a publisher creates an event, the broker will store the event until the subscriber becomes online and the event can be delivered.

Pub/sub is an open communication model, however, in many cases it may be desirable to protect the content of publications and subscriptions from unauthorized accesses. Only intended subscribers should be able to read the events. At the same time, subscribers may wish to keep the details of their filters private. For example, a subscriber may ask to be notified when the price of the quotes of a certain company is below a certain threshold. This information could reveal the subscriber's strategy to a competitor, thus the subscriber will wish to keep it private.

One of the main challenges that pub/sub systems are still facing is protecting the confidentiality of the exchanged information without limiting the decoupling of the paradigm. Publishers and subscribers do not establish contact so they cannot exchange keying material. Moreover, protecting the confidentiality from malicious brokers is very difficult. Brokers should be able to route events by matching them against filters expressed by the subscribers without having access to the actual content of events and filters.

Current solutions for confidentiality in pub/sub systems achieve only partially these goals. For example, in order to support routing based on expressive filters, [12] and [14] encrypt only certain event fields while other fields are left as cleartext so that they can be used for routing. Other solutions [14] require publishers and subscribers to share a group key which hampers the loosely coupling and scalability of pub/sub model. [16] provides confidentiality of events and filters but the filter is restricted to equality with one keyword.

The main contribution of this paper is to present an approach catering for the confidentiality in pub/sub systems such that: (i) it provides confidentiality of events and filters, (ii) it does not require publishers and subscribers to share keys, and (iii) it allows subscribers to express filters that can define any monotonic and non-monotonic conditions. To achieve this, our solution combines attribute-based encryption and an encrypted search scheme.

This paper is structured as follows: Section 2 introduces the pub/sub communication model and provides an example of an application where pub/sub confidentiality is required. Section 3 describes the problem of confidentiality and the properties achieved by our solution and Section 4 introduces the relevant encryption mechanisms. The details of our solution are provided in Section 5.

Section 6 revises the application example described in Section 2 implemented with our approach. Section 7 provides the security analysis. Section 8 describes the related work and Section 9 concludes the paper.

2 The Publish/Subscribe Communication Paradigm

Several pub/sub implementations that differ in the granularity used in the definition of the filters have been proposed in the literature. The most simple one is *topic-based*, in which subscribers subscribe to a topic identified by a *keyword* [20]. A topic-based scheme is similar to the notion of group communication. When subscribing to a topic T, a subscriber becomes a member of group T. When an event for topic T is published, the event is broadcasted to all the members of that group. Organizing topics in hierarchies allows a better management of subscriptions [17]. For example, by registering to a topic, a subscriber is also registered to all subtopics.

Topic-based schemes are easy to implement but they offer limited expressiveness. *Content-based* schemes are more flexible and allow expressing subscriptions based on the actual content of the event. To express a filter on the content of an event, subscribers need a query language and understanding of the data formats. For example, in Gryphon [2] and Siena [5] events consist of sets of $(attribute_{name} = attribute_{value})$ pairs and filters are specified as SQL WHERE clauses. Java Message Service (JMS) [11] does not allow filtering on the content of the event, but instead, events carry properties in their headers and subscribers can define filters on them. Filters that apply to the composition of simple events have also been proposed (such as in [1]). When expressing such a filter, subscribers are notified upon the occurrence of the composite event.

Because of its generality and expressiveness, we will focus on content-based filtering. We assume that filters define constrains in the form of *name-op-value* where *op* can be one of the comparison operators such as $=, \leq, <, \geq, >$. Constrains can be logically combined using AND, OR and NOT to form complex subscription patterns.

We motivate the need for confidentiality in pub/sub systems through an example.

2.1 A Case for Pub/Sub Confidentiality

In this section we present an example of an application built using a pub/sub system where confidentiality is of paramount importance. In particular, Figure 1 shows an example of a Financial News Service implemented using a pub/sub system for information delivery. The publishers P are different stock exchanges and financial news agencies which use the Financial News Service to sell their content to customers S. To subscribe to particular content, a customer specifies a filter and contacts the News Service to pay the fee and obtain a token. It then subscribes with a broker B to receive notifications and the broker registers the filter only after the token is verified with the Financial News Service. When a

publisher publishes some new content, the network of brokers will deliver the content to the authorized subscribers. The publisher receives the payment from the Financial News Service without contacting the subscribers directly.

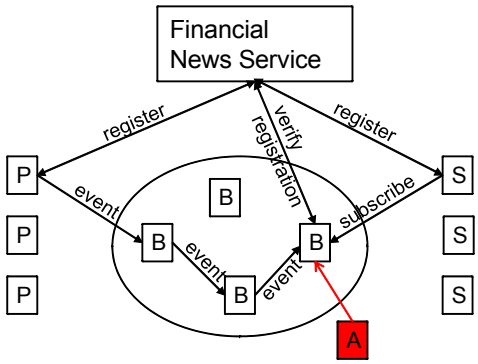

Fig. 1. An attacker who is able to corrupt a broker can listen on filters and events.

In a typical pub/sub system where confidentiality is not implemented, an attacker who is able to corrupt a broker could read the traffic that comes in and out the broker. The attacker would be able to read the events without paying the fee and then resell them, and read the filters expressed by the subscribers. To protect from this kind of attacks, it is necessary to protect the content of notifications and filters.

3 Confidentiality in Publish/Subscribe Systems

Providing the *publication confidentiality* property ensures that the content of the events is hidden from the broker or any unauthorized third party listening on the network. Only legitimate subscribers should be able to decrypt an event. Providing the *subscription confidentiality* property ensures that the details of the filters are hidden from the brokers (or other unauthorized parties). The broker should be able only to tell if an event matches a filter but gain no other information about the event and the filter. It has already been discussed in [16] that both publication and subscription confidentiality are required to effectively reduce the risk of leaking event or filter information in a pub/sub system. For instance, in providing only subscription confidentiality an attacker who knows the content of the event may infer the subscription filter.

However, providing both publication and subscription confidentiality in pub/sub systems it is still an open issue. On the one hand, a basic encryption scheme would require publisher and subscribers to share a secret key. This is not desirable

because it would weaken the referential decoupling property of the paradigm. On the other hand, brokers would need to execute matching operations on encrypted events and filters which is not simple using basic techniques.

The main contribution of this paper is to propose an encryption scheme for pub/sub systems in which the following properties are supported:

(P1) confidentiality of events;

(P2) confidentiality of filters;

(P3) a simplified key management that does not require publishers and subscribers to share keys, hence fully supporting the loosely-coupled model of the pub/sub paradigm;

(P4) allowing brokers to execute matching of encrypted events against complex encrypted filters. [3]

Confidentiality of events (P1) and filters (P2) can be achieved by means of encryption. Encryption mechanisms usually require that publishers and subscribers share a key which means they need to establish contact. However, this is not desirable in pub/sub systems where publisher and subscribers do not communicate with each other directly (loose coupling). What is required is a mechanism that allows authorized subscribers to decrypt events without establishing shared keys with the publishers (e.g., group keys). In our approach, publishers encrypt the content of the event using an attribute-based encryption scheme (such as in [10]) specifying the characteristics that subscribes must satisfy to obtain the cleartext of the event. In this way, we are effectively decoupling the encryption of events at the publisher site from its decryption at the subscriber site and simplifying the key management process (P3).

Because events and filters are encrypted, event filtering at the broker side becomes a more complex task. Indeed, brokers should be able to decide whether an event matches a filter or not, without having access to neither the content of the event nor the filter. In our approach we combine the expressive access control structures supported by attribute-base encryption scheme with encrypted search. This allows our scheme to support encrypted event filtering against complex filters. The only information that the broker can access is which filters are matched by an event (P4).

In the following section, we describe the techniques used in our approach for supporting the above properties.

4 Background

This section provides background information on the techniques that we have combined to achieve confidentiality in pub/sub systems without compromising the loosely-coupled property of the paradigm.

[3] With complex encrypted filters we mean filters that can express conjunctions and disjunctions of equalities, inequalities and negations in an encrypted form.

4.1 Attribute-based Encryption (ABE)

The concept of attribute-based encryption (ABE) was first introduced in [15]. In their construction, both ciphertext and keys are labeled with sets of attributes. A key is able to decrypt a ciphertext if at least k attributes match between key and ciphertext.

[10] extended this construction and introduced Key-Policy ABE (KP-ABE) in which ciphertexts are labelled with sets of attributes and private keys are associated with access structures. A key is able to decrypt a ciphertext if its associated access structure is satisfied by the attributes of the ciphertext. The access structure, represented as a tree, allows expressing any monotone access formula consisting of AND, OR, or threshold gates.

[13] proposed a KP-ABE scheme that can additionally handle negations (i.e., NOT). The data can be decrypted only if a given attribute (embedded in the key) is *not* present among the attributes of the ciphertext.

[4] proposed a construction for ciphertext policy ABE (CP-ABE) in which policies (access structures) are associated with data and attributes are associated with keys. This is similar to the capability model in access control. A key can decrypt some data if its associated attributes satisfy the policy associated with the data. They also show how to construct the access tree in order to additionally handle inequalities.

4.2 Encrypted Search

[18] proposed a mechanism for equality tests on data encrypted with a symmetric key. The advantages are that the searched keyword remains secret and the server cannot learn anything more about the data than the search results. However, the scheme works only for matching single words. The solution of [9] addresses the problem of conjunctions. Documents are stored encrypted together with a list of keywords, also encrypted. To retrieve a document, the user computes a capability for the list of keywords of interest. The server uses the capability to search for documents. The disadvantage of this method is that the server can learn the keywords from the capabilities.

[7] propose a data encryption scheme that allows an untrusted server to perform encrypted searches on data without revealing the data or the keywords to the server. The advantage of this method is that it allows multi-user access without the need for a shared key between users. Each user in the system has a unique set of keys. The data encrypted by one user can be decrypted by any other authorized user. The scheme is built on top of proxy encryption schemes. The idea is that a user defines a set of keywords for each document. The keywords and document are encrypted using proxy encryption and stored on the server. When a user wants to search for a document, it needs to create a trapdoor for each keyword. The trapdoor is used by the server to match the search keywords against the keywords of the stored document. The server can identify a match without learning the keyword.

5 Solution Details

In this section, we discuss in details our scheme for providing confidentiality in pub/sub systems. We assume an honest-but-curious model for publishers, brokers and subscribers, as in [19, 16]. This means that the entities follow the protocol, but may be curious to find out information by analysing the messages that are exchanged. For example, a broker may try to read the content of an event or try to learn the filtering constrains of subscribers. Subscribers may want to read the events delivered to other subscribers. We also assume that a passive attacker outside the pub/sub system may be able to listen on the communication and invade the privacy of the participants.

In our approach an event E consists of: (i) the message M that represents the content of the event and (ii) a set of attributes a_i that characterise M and are used for event filtering by the brokers.

To support confidentiality of events (P1), the message M is encrypted under the set of attributes a_i using KP-ABE [10]. In this way, only subscribers that expressed a filter that is satisfied by the attributes of the event, can decrypt the event. To allow publishers to specify additional constrains such as the attributes that a subscriber must have (e.g., age¿18), publishers can first encrypt event with CP-ABE [4]. In using CP-ABE and KP-ABE to encrypt M, publishers and subscribers do not need to share any secret key (thus achieving property P4).

Filter confidentiality (P2) is achieved by combining KP-ABE [10] with multi-user searchable data encryption (SDE) scheme [7]. In particular, a subscriber S_j can define a filter F_j as KP-ABE access trees. The set of attributes a_i that the publisher defined on an event E is used by the brokers against the filters. When the event E reaches a broker, if the set of attributes associated with the event satisfy the filter F_j, then the broker knows that the event can be forwarded to S_j.

However, if we encrypt the event and express the access tree as in the KP-ABE scheme proposed in [10], then the broker is still able to obtain information on the filters and attributes associated with events, thus violating the confidentiality of events and filters. In fact, the KP-ABE scheme requires that attributes associated with the ciphertext and the access tree are not encrypted. To circumvent this limitation, we propose the following modification to the KP-ABE scheme: the set of attributes associated with an event and the access tree representing the filter are encrypted using the scheme from [7]. The scheme supports encrypted search, so it can be used to verify if the encrypted attributes specified by the publisher are the same as those specified by the subscriber in the filter. With this modification, our scheme supports confidentiality of filters (P2) and allows the brokers to perform encrypted event filtering (P4). It should be noted that both KP-ABE and the multi-user SDE do not require that publishers and subscribers share keys thus simplifying the key management and respecting the referential decoupling of the pub/sub paradigm (P3).

In the following, we show the steps that are performed in our scheme.

5.1 Init(1^k)

The initialisation is run by a trusted authority and defines the security parameters for KP-ABE and El Gamal based SDE schemes.

On input 1^k, output two prime numbers p and q such that $q = (p-1)/2$ and $|q| = k$, and a cyclic group G_1 with generator g such that G_1 is the unique order q subgroup of $\mathbb{Z}_p^*$. Let $e : G_1$ x $G_1 \rightarrow G_2$ be a bilinear map. In addition, define the Lagrange coefficient $\Delta_{i,S}$ for $i \in \mathbb{Z}_p$ and a set S of elements in $\mathbb{Z}_p$: $\Delta_{i,S}(x) = \prod_{j \in S, j \neq i} \frac{x-j}{i-j}$. Each attribute will be mapped to a number in $\mathbb{Z}_p^*$ by using a collision resistant function $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$. This allows using arbitrary strings as attributes and adding them to a user's private key. The event will be encrypted using a set of n^4 elements of $\mathbb{Z}_p^*$.

Choose a random $y \in \mathbb{Z}_p$ and compute $g_1 = g^y$. Also choose a random element g_2 from G_1. Let N be the set $\{1, 2..., n+1\}$, where n is the number of attributes used for event encryption. Choose $t_1, ..., t_{n+1}$ uniformly at random from G_1. Define a function T as:

$$T(X) = g_2^{X^n} \prod_{i=1}^{n+1} t_i^{\Delta_{i,N}(X)}.$$

Publish the public parameters as: $PK_{KP} : g_1, g_2, t_1, ..., t_n$, and keep securely the master key $MK_{KP} : y$.

We define the parameters for the El Gamal based SDE scheme in group G_1 as in [7]. Let x be chosen uniformly at random from $\mathbb{Z}_p^*$ and compute $h = g^x$. Let H be a collision resistant hash function, f a pseudorandom function and s_1 a random key for f. Output the public and secret parameters for El Gamal based SDE: publish $PK_{SE} = (G_1, g, p, h, H, f)$, and keep securely $MK_{SE} = (x, s_1)$.

For every user (publisher or subscriber), run $Keygen(MK_{SE}, i)$ as in SDE, where i is the identity of the user. This function chooses x_{i1} random from $\mathbb{Z}_p$ and gives it to the user (publisher or subscriber) and computes $x_{i2} = x - x_{i1}$ and gives to the broker connected to the user the key (i,x_{i2}).

5.2 Event Encryption

Figure 2 shows the steps needed to encrypt an event. The publisher specifies a set of attributes γ under which the content $M \in G_2$ of the event will be encrypted.

Step 1. To provide confidentiality of M by expressing additional conditions that the subscriber must satisfy, the publisher encrypts M using the CP-ABE scheme. We call the message encrypted in this way M'.

Step 2. The publisher encrypts the message M' under γ as in KP-ABE. Choose a random $s \in \mathbb{Z}_p$ and compute the ciphertext as:

[4] with minor modifications, KP-ABE can encrypt to all sets of size $\leq n$.

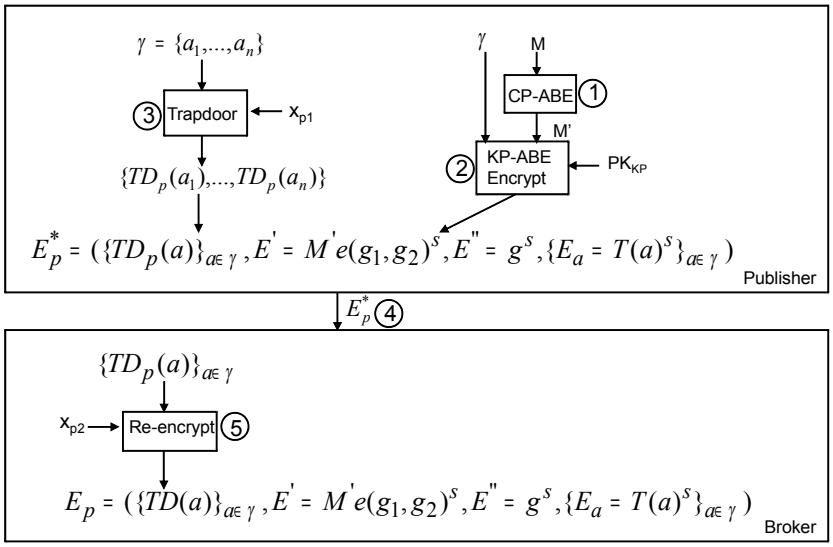

Fig. 2. Event encryption.

Encrypt(M', γ, PK_{KP})= (γ, E'=$M'e(g_1, g_2)^s$, E"=g^s, $\{E_a = T(a)^s)\}_{a\in\gamma}$)

Step 3. To provide confidentiality of attributes, the publisher encrypts them using multi-user SDE. For every attribute $a \in \gamma$, the publisher computes a trapdoor by calling Trapdoor((x_{p1}, s_1), a) as in multi-user SDE . Trapdoor() chooses a random r in $\mathbb{Z}_q$ and computes $TD_p(a) = (td_1, td_2)$ for each attribute such that $td_1 = g^{-r}g^{\sigma_a}$ and $td_2 = h^r g^{-x_{p1}r} g^{x_{p1}\sigma_a} = g^{x_{p2}r} g^{x_{p1}\sigma_a}$ where $\sigma_a = f_{s_1}(a)$.

Step 4. The publisher sends the encrypted event E^* together with the trapdoors for matching event attributes to the broker:

$$E^*_p=(\{TD_p(a)\}_{a\in\gamma}, \text{E'}=M'e(g_1, g_2)^s, \text{E"}=g^s, \{E_a = T(a)^s)\}_{a\in\gamma})$$

Note that we replaced the unencrypted set of attributes γ (as it appears in KP-ABE) with the encrypted trapdoor values $\{TD_p(a)\}_{a\in\gamma}$.

Step 4. The broker locates the key (p, x_{p2}) corresponding to the publisher and re-encrypts the trapdoors $\{TD_p(a)\}_{a\in\gamma}$. For each trapdoor $TD_p(a) = (td_1, td_2)$ it computes $TD(a) = td_1^{x_{p2}} td_2 = g^{x\sigma_a}$. The final encrypted event is:

$$E_p=(\{TD(a)\}_{a\in\gamma}, \text{E'}=M'e(g_1, g_2)^s, \text{E"}=g^s, \{E_a = T(a)^s)\}_{a\in\gamma}).$$

The above operations provide confidentiality of the message and attributes for an event, thus achieving property P1.

5.3 Filter Generation

Figure 3 shows the main steps for generating and encrypting the filter.

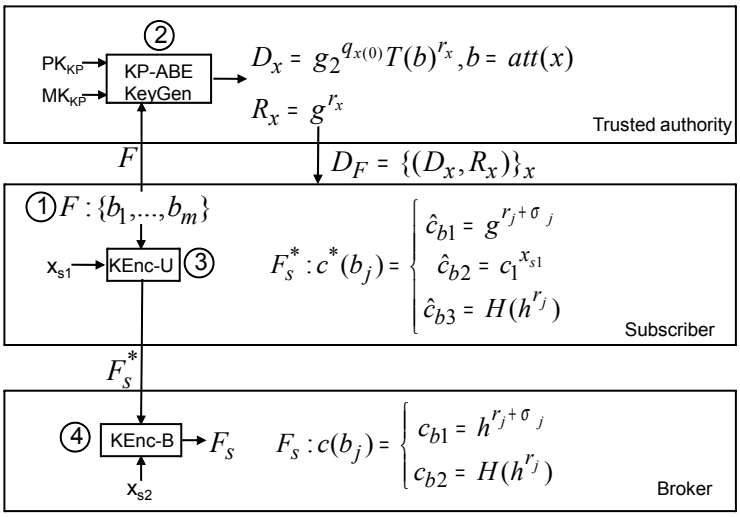

Fig. 3. Filter generation and encryption.

Step 1. The subscriber defines the filter as an access tree F. Each non-leaf node of the tree represents a threshold gate described by a value and its children. Let x be a node with num_x children. The threshold value k_x represents the number of children subtrees that need to be satisfied, hence $1 \leq k_x \leq num_x$. When $k_x = 1$ the threshold gate is an OR and when $k_x = num_x$, the threshold gate is an AND. Each leaf node x is described by an attribute and a threshold value $k_x = 1$.

We additionally define the following functions on the tree: parent(x) returns the parent of a node x and att(x) is defined only for a leaf node and returns the attribute associated with x. Further, we define an ordering between the children of every node x and give each child an index from 1 to num_x. The function index(x) returns the index associated to node x.

Step 2. As in KP-ABE, the subscriber sends the filter F to a trusted authority and requests a decryption key D_F. When applying to an event the filter D_F, the result will be a secret value that allows decrypting M' only if the attributes associated with the event match the filter. Otherwise the returned value will be a null value ($\perp$).

Choose a polynomial q_x for each node x in the tree F_s^*. The polynomials are chosen in a top down manner, starting from the root node r. For each node x in the tree, set the degree d_x of the polynomial q_x to be one less than the threshold value k_x of that node, that is, $d_x = k_x - 1$. Now for the root node r, set $q_r(0) = y$ and d_r other points of the polynomial q_r randomly to define it completely. For

any other node x, set $q_x(0) = q_{parent(x)}(index(x))$ and choose d_x other points randomly to completely define q_x.

Once the polynomials have been decided, for each leaf node x, the authority gives the following secret values to the subscriber:

$D_x = g_2^{q_x(0) \cdot T(b)^{r_x}}$, where b=att(x)

$R_x = g^{r_x}$

where r_x is chosen uniformly at random from $\mathbb{Z}_p$ for each node x. The set of the above values is the filter D_F, corresponding to a decryption key in KP-ABE.

Step 3. To provide confidentiality of the filter, the subscriber encrypts the leaf nodes using multi-user SDE. For every leaf node x in F run KEnc-U(x_{s1}, b), where b=att(x). Choose r at random from $\mathbb{Z}_p$ and compute $c^*(b) = (\hat{c}_{b1}, \hat{c}_{b2}, \hat{c}_{b3})$ where $\hat{c}_{b1} = g^{r+\sigma}$, $\sigma = f_{s_1}(b)$, $c_{b2} = \hat{c}_{b1}^{x_{s1}}$, $\hat{c}_{b3} = H(h^r)$.

Step 4. The subscriber sends F_s^* to the broker and keeps D_F securely. The broker locates the key (s_1, x_{s2}) corresponding to the subscriber and re-encrypts the leaf-node attributes of F_s^*. For each attribute $c^*(b)$ run KEnc-B$(s_1, x_{s2}, c^*(b))$. First compute $c(b) = (c_{b1}, c_{b2})$ such that $c_{b1} = \hat{c}_{b1}^{x_{s2}} \hat{c}_{b2} = \hat{c}_{b1}^{x_{s2}+x_{s1}} = (g^{r+\sigma})^x = h^{r+\sigma}$ where $\sigma = f_{s_1}(b)$ and $c_{b2} = \hat{c}_{b3}$.

The above operations provide confidentiality of the filter, thus achieving property P2. At the same time, the filter is able to express any access formula. We only give the details for expressing any monotone access formula consisting of AND, OR, or threshold gates, bur by extending the construction as in [13] and [4] we are able to represent inequalities and non-monotone access structures, thus achieving property P3.

5.4 Filtering of Events

When a new event E_p is published, for every filter F_s the broker runs a recursive algorithm on the tree F_s starting with the root node to check if it is satisfied by the attributes of the event. A non-leaf node x is satisfied if the number of satisfied children is equal of greater then k_x, the threshold value of the node. A leaf node $c(b) = (c_{b1}, c_{b2})$ is satisfied if the attribute contained by the node is among the attributes $TD(a)$ of the event. To check if a leaf node attribute b matches an event attribute a, the broker needs to verify if $c_{b2} = H(c_{b1} TD(a)^{-1})$. If the equality holds, the two attributes are the same. If not, there is no match. If the filter F_s is satisfied, the broker forwards the event following event E_p to the subscriber s s.

5.5 Decryption of the Content

Step 1. The subscriber performs the KP-ABE decryption of the event using the key D_F. The subscriber calls a recursive function $DecryptNode(E_p, D_F, x)$ on the root node of the tree F. If x is a leaf node, the function first checks if $att(x) = b$ encrypted as (c_{b1}, c_{b2}) is contained in the set of attributes of the event. For

every attribute a in the event encrypted as $TD(a)$, checks if $c_{b2} = H(c_{b1} \cdot TD^{-1})$. If this is the case, compute:

$$DecryptNode(E_p, D_F, x) = \frac{e(D_x, E")}{e(R_x, E_b)} = \frac{e(g_2^{q_x(0)} \cdot T(b)^{r_x}, g^s)}{e(g^{r_x}, T(b)^s)} = \frac{e(g_2^{q_x(0)}, g^s) \cdot e(T(b)^{r_x}, g^s)}{e(g^{r_x}, T(b)^s)} = e(g, g_2)^{sq_x(0)}$$

otherwise, $DecryptNode(E, D_F, x) = \perp$.

We now consider the recursive case when x is a non-leaf node. The algorithm $DecryptNode(E, D_F, x)$ then proceeds as follows. For all nodes z that are children of x, it calls $DecryptNode(E, D_F, z)$ and stores the output as F_z. Let S_x be an arbitrary k_x-sized set of child nodes z such that $F_z \neq \perp$. If no such set exists then the node was not satisfied and the function returns $\perp$. Otherwise, we compute:

$$F_x = \prod_{z \in S_x} F_z^{\Delta_{i,S'_x(0)}} \begin{cases} \text{where } i = index(z), \\ S' = \{index(z) : z \in S_x\} \end{cases}$$

$$= \prod_{z \in S_x} (e(g, g_2)^{s \cdot q_z(0)})^{\Delta_{i,S'_x}(0)}$$

$$= \prod_{z \in S_x} (e(g, g_2)^{s \cdot q_{parent(z)}(index(z))})^{\Delta_{i,S'_x}(0)} \text{ (by construction)}$$

$$= \prod_{z \in S_x} (e(g, g_2)^{s \cdot q_x(0)})^{\Delta_{i,S'_x}(0)} = e(g, g_2)^{sq_x(0)} \text{ (using polynomial interpolation)} (1)$$

and return the result. In case of a successful match, the subscriber obtains M' from $E' = M' e(g_1, g_2)^s$ by dividing it with $e(g_1, g_2)^s$.

If the publisher specified additional requirements by means of CP-ABE, the subscriber can decrypt the content M only if it holds the required attributes. It should be stressed that although publishers select the attributes, they do not know the subscribers. Publishers are only characterizing the subscribers so it could be the case that a subscriber who receives the event is not able to decrypt the content because it does not satisfy the properties specified by the publisher. For example, a publisher may want to send an event only to people belonging to a particular organization. Subscribers interested in the information but not belonging to that organization will not be able to decrypt the event.

6 Revisiting the Stock Quote Example

In the following we show how the example in Section 2.1 can be extended with the solution described above to provide confidentiality of events and filters.

As part of the initialization (see Section 5.1), the Trusted Authority generates the public(PK) and master (MK) keys for KP-ABE, CP-ABE and SDE. The public keys are published while the master keys are kept securely.

In our example, publisher P and subscriber S register with the Financial News Service. The Service contacts the Trusted Authority to generate the secret keys of the publisher and subscriber that will be used for SDE. The Trusted

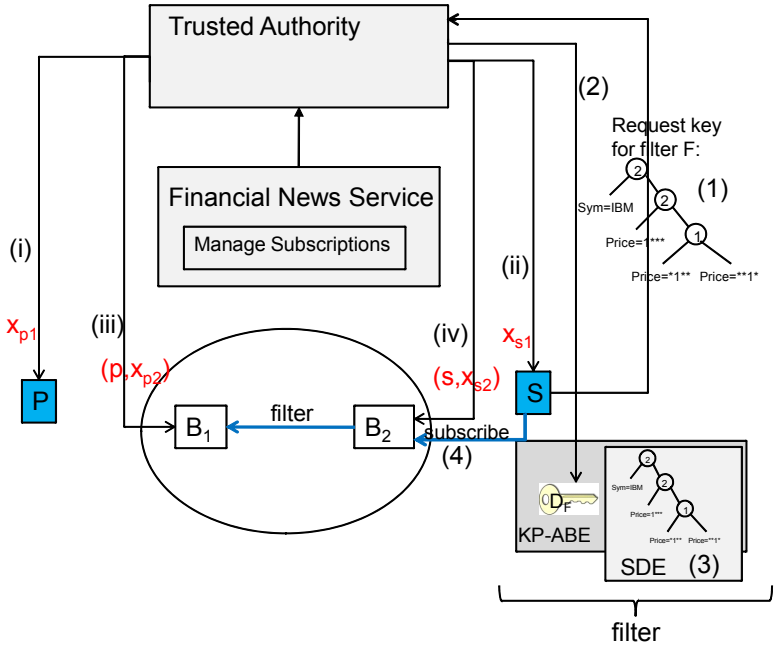

Fig. 4. Key and message exchange for filter generation.

Authority sends these keys on a secure channel to the publisher (i), subscriber (ii) and also to the brokers (iii, iv). These steps are shown in Figure 4.

Subscriber S expresses the subscription filter: "Sym=IBM" AND "Price>10". The following operations need to be performed (see Figure 4):

1. Construct the access tree corresponding to the filter. The tree representing the filter is shown in Figure 5. To represent the inequality "Price>10" we use the representation introduced in [4] and construct the access tree by expressing conditions on the bit values of the attribute. The threshold values of the nodes represent the number of sub-trees that need to be satisfied. In our example, 2 corresponds to an AND and 1 to an OR.
2. The subscriber sends this filter to the Trusted Authority which will generate a key D_F. This key is able to decrypt any event whose attributes satisfy the filter.
3. To ensure confidentiality of the filter, the attributes expressed in the leaf nodes are encrypted using SDE.
4. The subscriber sends the filter encrypted with SDE to the broker B_2 and keeps D_F securely. The broker re-encrypts the filter (i.e. the leaf nodes) and further distributes it in the pub/sub network.

Next, publisher P generates an event with the following attributes: "Sym=IBM" AND "Price=11". This event is to be received only by subscribers with the

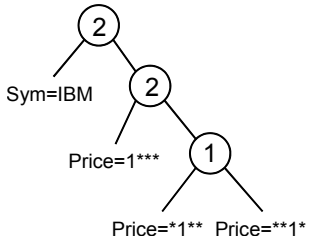

Fig. 5. Access tree for "Sym=IBM" AND "Price>10".

attribute "Premium customer". The publisher performs the following operations, as shown in Figure 6.

1. Encrypt the message content M with CP-ABE under the access policy "Premium customer". To do this, the publisher only needs the public parameters for CP-ABE distributed by the trusted authority at the beginning. This encryption ensure that only subscribers who possess the attribute "Premium customer" will be able to read the message.
2. To allow comparisons of numerical values, the publisher creates an attribute for each bit of the numerical attribute (as introduced in [4]). For Price=11 (1011), the attributes are: Price=1***, Price=*0**, Price=**1* and Price=***1.
3. Encrypt the message M using KP-ABE under the defined attributes.
4. To provide confidentiality of the attributes, encrypt all attributes using the SDE *Trapdoor()* function(see section 5.2).
5. Send to broker B_1 M encrypted under CP-ABE and KP-ABE and the encrypted attributes as trapdoors computed as in SDE.

Broker B_1 re-encrypts the attribute trapdoors of the received event and matches it against the stored filters. It then forwards the event to broker B_2. Broker B_2 matches the event against the filter from subscriber S and forwards the event to S. Subscriber S decrypts the event using D_F, the KP-ABE decryption key corresponding to its filter. Finally, assuming that S had the attribute "Premium customer", S is able to decrypt the message using CP-ABE.

7 Security Analysis

This section evaluates the security of the scheme. To ensure confidentiality of events our scheme encrypts both messages and associated attributes to prevent attackers to infer an event from its attributes. Messages are encrypted using CP-ABE encryption [4] and KP-ABE encryption [10] with non-monotonic filters; the attributes are encrypted using the multi-user SDE scheme.

All the used encryption schemes are proved to be at least indistinguishable under chosen plaintext attack (*IND-CPA*). [6] proves CP-ABE to be chosen plaintext (CPA) secure under the Decisional bilinear Diffie-Hellmann (DBDH) assumption, generally considered a hard problem. About multi-user SDE [7]

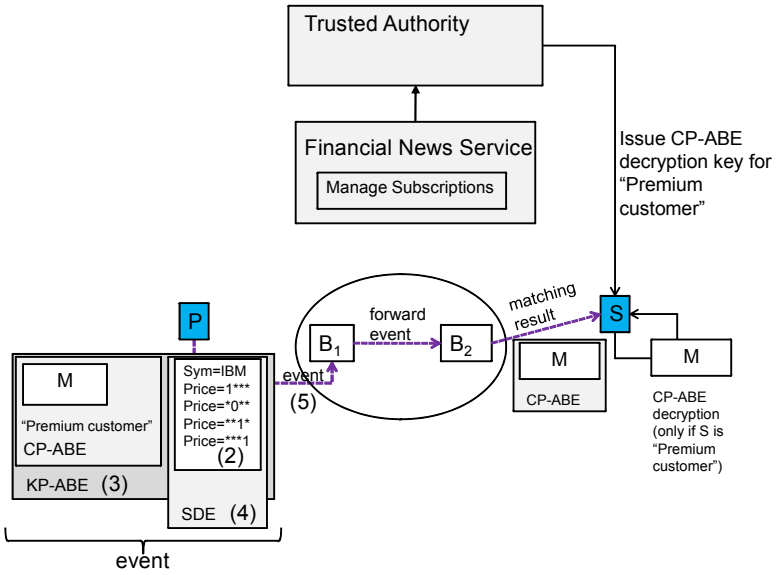

Fig. 6. Key and message exchange for event generation.

proves that the concrete construction, our scheme uses, built upon El Gamal-based proxy encryption is indistinguishable under chosen plaintext attack (*IND-CPA*) under the assumption the Decisional Diffie-Hellmann problem is hard relative to the group on which El Gamal is defined. About the KP-ABE scheme, [13] proves that the IND-CPA security of KP-ABE with non-monotonic access structures in the attribute-based selective-set model reduces to the hardness of the Decisional bilinear Diffie-Hellmann (DBDH) assumption, generally considered a hard problem. Hence the encryption scheme that protects events is IND-CPA secure.

All encryption primitives used by our scheme are *IND-CPA* secure, what is left is to show is their combination is still secure. The different mechanisms are used as multiple layer of encryption and [3] shows that if a cryptosystem is secure in the sense of indistinguishability, then the cryptosystem in the multi-user setting, where related messages are encrypted using different keys, is also secure. In our case each encryption layer uses an independent key so the combination is at least as secure as any individual encryption. Thus, the scheme is at least *IND-CPA* secure.

Filters' confidentiality is achieved by encrypting KP-ABE access tress with multi-user SDE. Thus, filter encryption is *IND-CPA* secure.

8 Related Work

Current solutions for ensuring confidentiality in publish/subscribe systems provide only some of the properties satisfied by our solution, but not all of them at the same time. For example, [12] proposes a scheme that does not require publishers and subscribers to share a key, but does not achieve full confidentiality of events and confidentiality of filters. Events are encoded in XML format, but only specific fields (e.g., price) are encrypted with a symmetric key k. The publisher then encrypts k with its public key. The brokers forward the event based on the fields left unencrypted and a proxy service changes the encryption of k to an encryption with the public key of the subscriber.

In [14] Raiciu and Rosenblum achieve partial confidentiality but they require that publishers and subscribers share a group key which is used to encrypt events and filters. In their model, notifications are composed of $(name, value)$ pairs where only value is encrypted which in some scenarios may not provide a sufficient level of confidentiality.

In [19], Srivartsa & Liu propose a specific key management scheme and a probabilistic multi-path event routing to prevent frequency inferring attacks. The method achieves confidentiality of events and filters, however, filtering is done based on only one keyword. A centralized trusted authority distributes encryption keys to publishers and authorization keys to subscribers. Inequalities are supported by using a hierarchical key structure where each key corresponds to an interval. However, the inequality condition cannot be checked by the brokers, instead, after receiving an event corresponding to the specified keyword, a subscriber will be able to decrypt it only if the numerical value of the event's attribute is in the range corresponding to the subscriber's authorization key.

In [16], Shikfa et al. propose a solution based on multiple layer commutative encryption that achieves content and filter confidentiality, and routing of encrypted data. The advantage of this method is that key management is local and publisher and subscribers do not need to share keys. However, the filter is limited to equality filter with only one keyword.

9 Conclusions and Future Work

In this paper, we presented a solution for providing confidentiality in pub/sub systems. Our solution is an encryption scheme based on CP-ABE, KP-ABE and multi-user SDE. Our scheme supports both the publication and the subscription confidentiality properties while at the same time does not require publishers and subscribers to share secret keys. Although events and filters are encrypted, brokers can still perform event filtering without learning any information. Finally, our scheme allows subscribers to express filters that can define any monotonic and non-monotonic constraints on events.

As future work, we are working on a more formal proof to evaluate the security of the scheme. At the same time, we are planning to implement our scheme and to include it in one of the mainstream implementations of the pub/sub

model. This would allow us to assess the impact in performance that such scheme imposes on the resources of the pub/sub system.

Acknowledgements

The work of the second author is supported by the EU project Consequence Research Grant FP7-214859. The work of the third author is partially funded by the EU project MASTER contract no. FP7-216917.

References

1. J. Bacon, K. Moody, J. Bates, R. Hayton, C. Ma, A. McNeil, O. Seidel, and M. Spiteri. Generic support for distributed applications. IEEE Computer, 33(3):68-76, 2000.
2. G. Banavar, T. Chandra, B. Mukherjee, J. Nagarajarao, R. Strom, and D. Sturman. An efficient multicast protocol for content-based publish-subscribe systems. In International Conference on Distributed Computing Systems, volume 19, pages 262-272. IEEE COMPUTER SOCIETY PRESS, 1999.
3. M. Bellare, A. Boldyreva, and J. Staddon. Multi-recipient encryption schemes: Security notions and randomness re-use. In PKC 2003: Public Key Cryptography, volume 2567. Springer-Verlag, 2003.
4. J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In IEEE Symposium on Security and Privacy, pages 321-334. Citeseer, 2007.
5. A. Carzaniga, D.S. Rosenblum, and A.L. Wolf. Design and evaluation of a wide-area event notification service. ACM Transactions on Computer Systems (TOCS), 19(3):332-383, 2001.
6. L. Cheung and C. Newport. Provably secure ciphertext policy abe. In CCS '07: Proceedings of the 14th ACM conference on Computer and communications security, pages 456-465, New York, NY, USA, 2007. ACM.
7. C. Dong, G. Russello, and N. Dulay. Shared and searchable encrypted data for untrusted servers. In Proceedings of the 22nd annual IFIP WG 11.3 working conference on Data and Applications Security, pages 127-143, Berlin, Heidelberg, 2008. Springer-Verlag.
8. P.T. Eugster, P.A. Felber, R. Guerraoui, and A.M. Kermarrec. The many faces of publish/subscribe. ACM Computing Surveys (CSUR), 35(2):131, 2003.
9. P. Golle, J. Staddon, and B. Waters. Secure conjunctive keyword search over encrypted data. Lecture notes in computer science, 3089:31-45, 2004.
10. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Proceedings of the 13th ACM conference on Computer and communications security, page 98. ACM, 2006.
11. R. Burridge, R. Sharma, J. Fialli, M. Hapner, and K. Stout. Java message service. Sun Microsystems Inc., Santa Clara, CA, 2002.
12. H. Khurana. Scalable security and accounting services for content-based publish/subscribe systems. In Proceedings of the 2005 ACM symposium on Applied computing, page 807. ACM, 2005.
13. R. Ostrovsky, A. Sahai, and B. Waters. Attribute-based encryption with non-monotonic access structures. In Proceedings of the 14th ACM conference on Computer and communications security, page 203. ACM, 2007.

14. C. Raiciu and D.S. Rosenblum. Enabling confidentiality in content-based publish/subscribe infrastructures. Securecomm and Workshops, 28:1-11, 2006.
15. A. Sahai and B. Waters. Fuzzy identity-based encryption. In EUROCRYPT, volume 3494, pages 457473. Springer, 2005.
16. A. Shikfa, M. Onen, and R. Molva. Privacy-Preserving Content-Based Publish/Subscribe Networks. In Emerging Challenges for Security, Privacy and Trust: 24th Ifip Tc 11 International Information Security Conference, SEC 2009, Pafos, Cyprus, May 18-20, 2009, Proceedings, page 270. Springer, 2009.
17. Z.U. Singhera. A workload model for topic-based publish/subscribe systems. 2008.
18. D.X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In 2000 IEEE Symposium on Security and Privacy, 2000. SP 2000. Proceedings, pages 44-55, 2000.
19. M. Srivatsa and L. Liu. Secure event dissemination in publish-subscribe networks. In Proceedings of the 27th International Conference on Distributed Computing Systems, page 22. Citeseer, 2007.
20. S.Q. Zhuang, B.Y. Zhao, A.D. Joseph, R.H. Katz, and J.D. Kubiatowicz. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video, page 20. ACM, 2001.