Securing Broker-Less Publish/Subscribe Systems Using Identity-Based Encryption

Muhammad Adnan Tariq, Boris Koldehofe, and Kurt Rothermel

Abstract—The provisioning of basic security mechanisms such as authentication and confidentiality is highly challenging in a content-based publish/subscribe system. Authentication of publishers and subscribers is difficult to achieve due to the loose coupling of publishers and subscribers. Likewise, confidentiality of events and subscriptions conflicts with content-based routing. This paper presents a novel approach to provide confidentiality and authentication in a broker-less content-based publish/subscribe system. The authentication of publishers and subscribers as well as confidentiality of events is ensured, by adapting the pairing-based cryptography mechanisms, to the needs of a publish/subscribe system. Furthermore, an algorithm to cluster subscribers according to their subscriptions preserves a weak notion of subscription confidentiality. In addition to our previous work [23], this paper contributes 1) use of searchable encryption to enable efficient routing of encrypted events, 2) multicredential routing a new event dissemination strategy to strengthen the weak subscription confidentiality, and 3) thorough analysis of different attacks on subscription confidentiality. The overall approach provides fine-grained key management and the cost for encryption, decryption, and routing is in the order of subscribed attributes. Moreover, the evaluations show that providing security is affordable w.r.t. 1) throughput of the proposed cryptographic primitives, and 2) delays incurred during the construction of the publish/subscribe overlay and the event dissemination.

Index Terms—Content-based, publish/subscribe, peer-to-peer, broker-less, security, identity-based encryption

1 Introduction

The publish/subscribe (pub/sub) communication paradigm has gained high popularity because of its inherent decoupling of publishers from subscribers in terms of time, space, and synchronization. Publishers inject information into the pub/sub system, and subscribers specify the events of interest by means of subscriptions. Published events are routed to their relevant subscribers, without the publishers knowing the relevant set of subscribers, or vice versa. This decoupling is traditionally ensured by intermediate routing over a broker network [10]. In more recent systems, publishers and subscribers organize themselves in a broker-less routing infrastructure, forming an event forwarding overlay [24].

Content-based pub/sub is the variant that provides the most expressive subscription model, where subscriptions define restrictions on the message content. Its expressiveness and asynchronous nature is particularly useful for large-scale distributed applications such as news distribution, stock exchange, environmental monitoring, traffic control, and public sensing. Not surprisingly, pub/sub needs to provide supportive mechanisms to fulfill the basic security demands of these applications such as access control and confidentiality.

• *The authors are with the Institute of Parallel and Distributed Systems, University of Stuttgart, Universitätsstraße 38, Stuttgart 70569, Germany. E-mail: {adnan.tariq, boris.koldehofe, kurt.rothermel}@ipvs.uni-stuttgart.de.*

Manuscript received 16 Sept. 2012; revised 23 Sept. 2013; accepted 24 Sept. 2013; published online 4 Oct. 2013.
Recommended for acceptance by X. Li, P. McDaniel, R. Poovendran, and G. Wang.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDSSI-2012-09-0915.
Digital Object Identifier no. 10.1109/TPDS.2013.256.

Access control in the context of pub/sub system means that only authenticated publishers are allowed to disseminate events in the network and only those events are delivered to authorized subscribers. Moreover, the content of events should not be exposed to the routing infrastructure and a subscriber should receive all relevant events without revealing its subscription to the system. Solving these security issues in a content-based pub/sub system imposes new challenges. For instance, end-to-end authentication using a *public key infrastructure (PKI)* conflicts with the loose coupling between publishers and subscribers, a key requirement for building scalable pub/sub systems. For PKI, publishers must maintain the public keys of *all* interested subscribers to encrypt events. Subscribers must know the public keys of *all* relevant publishers to verify the authenticity of the received events. Furthermore, traditional mechanisms to provide confidentiality by encrypting the whole event message conflict with the content-based routing paradigm. Hence, new mechanisms are needed to route encrypted events to subscribers without knowing their subscriptions and to allow subscribers and publishers authenticate each other without knowing each other.

In the past, most research has focused only on providing expressive and scalable pub/sub systems, but little attention has been paid for the need of security. Existing approaches toward secure pub/sub systems mostly rely on the presence of a traditional broker network [20], [2], [9], [22], [7], [18], [16]. These either address security under restricted expressiveness, for example, by using only keyword matching for routing events [22], [21] or rely on a network of (semi-)trusted brokers [19], [17], [12]. Furthermore, existing approaches use coarse-grain epoch-based key management and cannot provide fine-grain access control in a scalable manner [22], [20]. Nevertheless,

security in broker-less pub/sub systems, where the subscribers are clustered according to their subscriptions, has not been discussed yet in the literature.

Building on our results of [23], this paper presents a new approach to provide authentication and confidentiality in a broker-less pub/sub system. Our approach allow subscribers to maintain credentials according to their subscriptions. Private keys assigned to the subscribers are labeled with the credentials. A publisher associates each encrypted event with a set of credentials. We adapted identity-based encryption (IBE) mechanisms [4], [8] 1) to ensure that a particular subscriber can decrypt an event only if there is a match between the credentials associated with the event and the key; and 2) to allow subscribers to verify the authenticity of received events. Furthermore, we address the issue of subscription confidentiality in the presence of semantic clustering of subscribers. A weaker notion of subscription confidentiality is defined and a secure overlay maintenance protocol is designed to preserve the weak subscription confidentiality.

In addition to [23], we also present 1) extensions of the cryptographic methods to provide efficient routing of encrypted events by using the idea of searchable encryption, 2) "Multicredential routing" a new event dissemination strategy which strengthens the weak subscription confidentiality, and 3) a thorough analysis of different attacks on subscription confidentiality. Moreover, the supplemental document, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2013.256, presents detailed analysis of the correctness of cryptographic methods used in this paper and a concise review of the related work.

2 System Model and Background

2.1 Content-Based Publish/Subscribe

For the routing of events from publishers to the relevant subscribers, we use the content-based data model. The *event space*, denoted by Ω, is composed of a global ordered set of d distinct attributes (A_i): $\Omega = \{A_1, A_2, \ldots, A_d\}$. Each *attribute* A_i is characterized by a unique *name*, its *data type*, and its *domain*. The data type can be any ordered type such as integer, floating point, and character strings. The domain describes the range $[L_i, U_i]$ of possible attribute values. A *subscription filter* f is a conjunction of predicates, i.e., $f = \{Pred_1 \wedge Pred_2 \cdots \wedge Pred_j\}$. $Pred_i$ is defined as a tuple (A_i, Op_i, v_i), where Op_i denotes an operator and v_i a value. The operator Op_i typically includes equality and range operations for numeric attributes and prefix/suffix operations for strings. An *event* consists of attributes and associated values. An event is *matched* against a subscription f if the values of attributes in the event satisfy the corresponding constraints imposed by the subscription.

We consider pub/sub in a setting where there exits no dedicated broker infrastructure. Publishers and subscribers contribute as peers to the maintenance of a self-organizing overlay structure. To authenticate publishers, we use the concept of *advertisements* in which a publisher announces beforehand the set of events which it intends to publish.

2.2 Attacker Model

Our attacker model is similar to the commonly used *honest-but-curious* model [22], [21]. There are two entities in the system: publishers and subscribers. Both the entities are computationally bounded and do not trust each other. Moreover, all the peers (publishers or subscribers) participating in the pub/sub overlay network are honest and do not deviate from the designed protocol. Likewise, authorized publishers only disseminate valid events in the system. However, malicious publishers may masquerade the authorized publishers and spam the overlay network with fake and duplicate events. We do not intend to solve the digital copyright problem; therefore, authorized subscribers do not reveal the content of successfully decrypted events to other subscribers.

Subscribers are, however, curious to discover the subscriptions of other subscribers and published events to which they are not authorized to subscribe. Similarly, curious publishers may be interested to read events published in the system. Furthermore, passive attackers outside the pub/sub overlay network can eavesdrop the communication and try to discover content of events and subscriptions.

Finally, we assume the presence of secure channels for the distribution of keys from the key server to the publishers and subscribers. A secure channel can be easily realized by using transport layer mechanisms such as *Transport Layer Security* (TLS) or *Secure Socket Layer* (SSL).

2.3 Security Goals and Requirements

There are three major goals for the proposed secure pub/sub system, namely to support authentication, confidentiality, and scalability.

Authentication. To avoid noneligible publications, only authorized publishers should be able to publish events in the system. Similarly, subscribers should only receive those messages to which they are authorized to subscribe.

Confidentiality. In a broker-less environment, two aspects of confidentiality are of interest: 1) the events are only visible to authorized subscribers and are protected from illegal modifications, and 2) the subscriptions of subscribers are confidential and unforgeable.

Scalability. The secure pub/sub system should scale with the number of subscribers in the system. Three aspects are important to preserve scalability: 1) the number of keys to be managed and the cost of subscription should be independent of the number of subscribers in the system, 2) the key server and subscribers should maintain small and constant numbers of keys per subscription, and 3) the overhead because of rekeying should be minimized without compromising the fine-grained access control.

2.4 Identity-Based Encryption

While a traditional PKI infrastructure requires to maintain for each publisher or subscriber a private/public key pair which has to be known between communicating entities to encrypt and decrypt messages, *identity-based encryption* [6] provides a promising alternative to reduce the amount of keys to be managed. In identity-based encryption, any valid string which uniquely identifies a user can be the public key of the user. A key server maintains a single pair of public

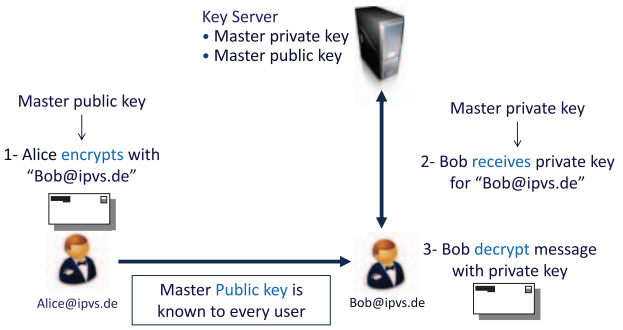


Fig. 1. Identity-based encryption.

and private master keys. The master public key can be used by the sender to encrypt and send the messages to a user with any identity, for example, an e-mail address. To successfully decrypt the message, a receiver needs to obtain a private key for its identity from the key server. Fig. 1 shows the basic idea of using identity-based encryption.

We want to stress here that although identity-based encryption at the first glance appears like a highly centralized solution, its properties are ideal for highly distributed applications. A sender needs to know only a single master public key to communicate with any identity. Similarly, a receiver only obtains private keys for its own identities. Furthermore, an instance of central key server can be easily replicated within the network. Finally, a key server maintains only a single pair of master keys and, therefore, can be realized as a smart card, provided to each participant of the system.

Although identity-based encryption has been proposed some time ago, only recently *pairing-based cryptography* (PBC) has laid the foundation of practical implementation of *identity-based* encryption. Pairing-based cryptography establishes a mapping between two cryptographic groups by means of *bilinear maps*. This allows the reduction of one problem in one group to a different usually easier problem in another group. We utilize bilinear maps for establishing the basic security mechanisms in the pub/sub system and, therefore, introduce here the main properties. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be cyclic group of order q, where q is some large prime. A bilinear map is a function $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ that associates a pair of elements from $\mathbb{G}_1$ to elements in $\mathbb{G}_2$. A bilinear map satisfies the following conditions:

1. *Bilinearity.* $\hat{e}(u^x, v^y) = \hat{e}(u^y, v^x) = \hat{e}(u, v)^{xy}$, for all $u, v \in \mathbb{G}_1$, and $x, y \in \mathbb{Z}$.
2. *Nondegeneracy.* $\hat{e}(u, v) \neq 1$, for all $u, v \in \mathbb{G}_1$.
3. *Computability.* $\hat{e}$ can be efficiently computed.

3 Approach Overview

For providing security mechanisms in pub/sub, we leverage the principles of identity-based encryption to support many-to-many interactions between subscribers and publishers. Although we subsequently demonstrate the implementation of our security methods in terms of a concrete variant called *attribute-based encryption*, it is important to remark that our approach also benefits from other identity-based encryption schemes.

In our approach, publishers and subscribers interact with a key server. They provide *credentials* to the key server and in turn receive keys which fit the expressed capabilities in the credentials. Subsequently, those keys can be used to encrypt, decrypt, and sign relevant messages in the content-based pub/sub system, i.e., the credential becomes *authorized* by the key server. A credential consists of two parts: 1) a binary string which describes the capability of a peer in publishing and receiving events, and 2) a proof of its identity. The latter is used for authentication against the key server and verification whether the capabilities match the identity of the peer. While this can happen in a variety of ways, for example, relying on challenge response, hardware support, and so on, we pay attention mainly at expressing the capabilities of a credential, i.e., how subscribers and publishers can *create* a credential. This process needs to account for the many possibilities to partition the set of events expressed by an advertisement or subscription and exploits overlaps in subscriptions and publications. Subsequently, we use the term credential only for referring to the capability string of a credential.

The keys assigned to publishers and subscribers, and the ciphertexts, are labeled with credentials. In particular, the identity-based encryption ensures that a particular key can decrypt a particular ciphertext only if there is a match between the credentials of the ciphertext and the key. Publishers and subscribers maintain separate private keys for each authorized credential.

The public keys are generated by a string concatenation of a credential, an epoch for key revocation, a symbol $\in \{SUB, PUB\}$ distinguishing publishers from subscribers, and some additional parameters described in Section 5. The public keys can be easily generated by any peer without contacting the key server or other peers in the system. Similarly, encryption of events and their verification using public keys do not require any interaction.

Due to the loose coupling between publishers and subscribers, a publisher does not know the set of relevant subscribers in the system. Therefore, a published event is encrypted with the public key of all possible credentials, which authorizes a subscriber to successfully decrypt the event. The ciphertexts of the encrypted event are then signed with the private key of the publisher, as shown in Fig. 2.

The overlay network is maintained according to the containment relationship between the subscriptions. Subscribers with coarser subscriptions are placed near the root and forward events to the subscribers with less coarser subscriptions. To maintain such a topology, each subscriber should know the subscription of its parent and child peers. When a new subscriber arrives, it sends the connection request (CR) along with its subscription to a random peer in the overlay network. The connection request is forwarded by possibly many peers in the overlay network before it reaches the right peer to connect. Each forwarding peer matches the subscription in the request with the subscription of its parent and child peers to decide the forwarding direction. Maintaining a relationship between subscriptions clearly contradicts subscription confidentiality. Therefore, we show the approach to ensure a weaker notion of subscription confidentiality in Section 6.

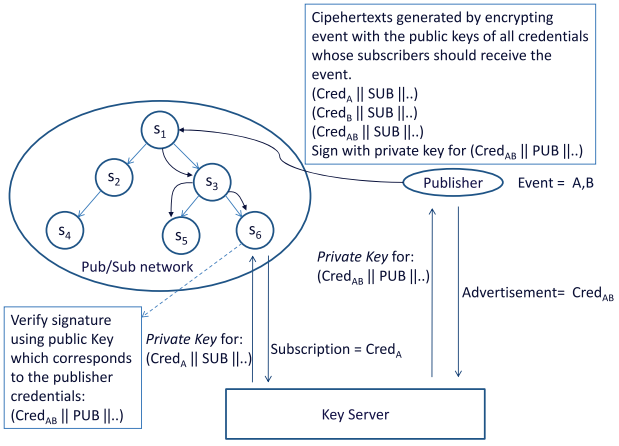


Fig. 2. Approach overview: Publisher has credentials to publish events with two attributes A and B. Subscriber s_6 has credentials to receive events with attribute A.

4 Creation of Credentials

In the following, we will first describe the creation of credentials for numeric and string attributes. Further extensions to handle complex subscriptions are discussed subsequently.

4.1 Numeric Attributes

The event space, composed of d distinct numeric attributes, can be geometrically modeled as a d-dimensional space such that each attribute represents a dimension in the space. With the spatial indexing approach, the event space is hierarchically decomposed into regular subspaces, which serve as enclosing approximation for the subscriptions, advertisements, and events. The decomposition procedure divides the domain of one dimension after the other and recursively starts over in the created subspaces. Fig. 3 visualizes the advancing decomposition with the aid of a binary tree.

Subspaces are identified by a bit string of "0" and "1"s. A subspace represented by dz_1 is covered by the subspace represented by dz_2, if dz_2 is a prefix of dz_1. Subscription or advertisement of a peer can be composed of several subspaces. A credential is assigned for each of the mapped subspace. For instance, in Fig. 3, f_2 is mapped to two subspaces and therefore possesses two credentials $\{000, 010\}$. An event can be approximated by the smallest

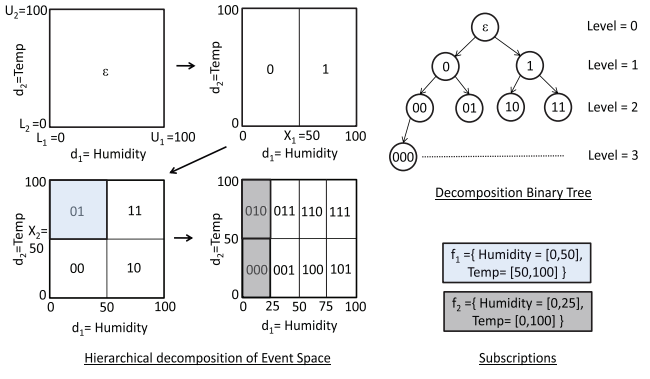


Fig. 3. Numeric attribute.

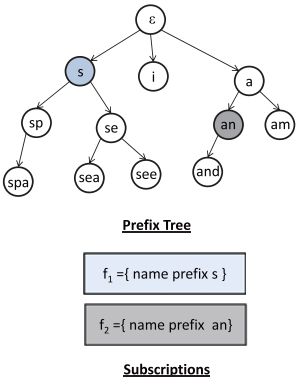


Fig. 4. Prefix matching.

(finest granularity) subspace that encloses the point represented by it. To deliver the encrypted event, a ciphertext must be generated for each subspace that encloses the event so that the peer whose subscription mapped to any of these subspaces should be able to successfully decrypt the event. For example, an event 0010 is enclosed by the five subspaces 0010, 001, 00, 0, and ϵ.

For an event space with a large set of numeric attributes, the number of mapped subspaces and, therefore, credentials for a subscription can be very large. This affects the scalability of the system. We address this problem by decomposing the domain of each attribute into subspaces separately. The spatial indexing procedure is the same as above; however, in this case, a separate decomposition tree is built for each attribute. Each peer receives credentials separately for each attribute in its subscription. The number of credentials maintained for each subscription or advertisement is bounded by $\sum_{i=1}^{d} \log_2(\mathcal{Z}_i)$, where $\mathcal{Z}_i = \frac{U_i - L_i}{Granularity(i)}$ and $Granularity(i)$ defines the smallest addressable value of the attribute A_i. Similarly, the number of subspaces matched by an event is $\sum_{i=1}^{d} \log_2(\mathcal{Z}_i)$.

4.2 String Attributes

The above spatial indexing technique can work with any ordered data type with a known domain. String attributes usually have a maximum number of characters. This allows them to have known bounds. They can be linearized by hashing or other linearization mechanisms and, thus, can also be indexed [24].

Credentials for more expressive string operations such as prefix matching can be generated using a *trie*. Each node in the trie is labeled with a string, which serves as a common prefix to all its descendants, as shown in Fig. 4. Each peer is assigned a single credential, which is same as its subscription or advertisement. Events correspond to the leaf nodes of the trie. To deliver an encrypted event, a ciphertext must be generated with the label of each node in the path from the leaf to the root of the trie, so that a peer whose subscription matches any of the labels should be able to successfully decrypt the event. In general, the number of nodes on the longest path from a leaf to the root of a trie associated with a string attribute A_i is equal to $\mathcal{L}_i$, where $\mathcal{L}_i$ is the length of the longest label assigned to a leaf node. Similar mechanism can be used to generate credentials for suffix matching.

4.3 Complex Subscriptions

For a complex subscription with predicates on different attributes, a subscriber receives separate credentials and, thus, keys for each attribute. Using these keys, a subscriber should be able to successfully decrypt any event with the corresponding attributes, if he is authorized to read the values associated with the attributes. In a content-based pub/sub system, a subscription defines a conjunction on predicates. An event matches a subscription if and only if all of the predicates in the subscription are satisfied. To ensure event confidentiality, a subscriber must not be able to successfully decrypt any event which matches only parts of its subscriptions. For example, consider a subscriber with two subscriptions $f_1 = \{Area = [10, 20] \wedge location = Stuttgart\}$ and $f_2 = \{Area = [40, 80] \wedge location = London\}$. If the credentials and, therefore, keys are assigned for individual attributes, then the subscriber can also decrypt the events matching the subscriptions $f_3 = \{Area = [10, 20] \wedge location = London\}$ and $f_4 = \{Area = [40, 80] \wedge location = Stuttgart\}$, although he is not authorized to read events matching the subscriptions f_3 and f_4. To properly ensure event confidentiality, all the keys associated with a subscription should be bound together, so that keys associated with different subscriptions should not be combined together.

5 Publisher/Subscriber Authentication and Event Confidentiality

The security methods describe in this section are built upon ciphertext-policy attribute-based encryption (in short CP-ABE) scheme proposed by Bethencourt et al. [4]. In particular, our modifications 1) allow publishers to sign and encrypt events at the same time by using the idea of the identity-based signcryption proposed by Yu et al. [25], 2) enable efficient routing of encrypted events (from publishers to subscribers) by using the idea of searchable encryption proposed by Boneh et al. [5], and 3) allow subscribers to verify the signatures associated with all the attributes (of an event) simultaneously. Our modifications do not change the basic structure of the CP-ABE scheme and preserves the same security strength, as discussed in the supplemental document available online.

5.1 Security Parameters and Initialization

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ denote the bilinear groups of prime order q, i.e., $|\mathbb{G}_1| = |\mathbb{G}_2| = q$, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ denote an admissible bilinear map, and g denote a generator in $\mathbb{G}_1$. Moreover, let $H_1 : \{0,1\}^* \rightarrow \{0,1\}^{n_u}$, $H_2 : \{0,1\}^* \rightarrow \{0,1\}^{n_m}$, $H_3 : \{0,1\}^* \rightarrow \mathbb{G}_1$, and $H_4 : \mathbb{G}_2 \rightarrow \{0,1\}^{\log q}$ designate collusion resistant cryptographic hash functions.

The initialization algorithm

1. chooses $\alpha, \varphi \in \mathbb{Z}_q$,
2. computes $g_1 = g^\alpha$ and $h = g^\varphi$,
3. chooses $g_2, u', m' \in \mathbb{G}_1$, and
4. selects vectors $\vec{u} = (u_i)$ and $\vec{m} = (m_i)$ of length n_u and n_m, respectively, with every element chosen uniformly at random from $\mathbb{G}_1$.

The *Master Public Key* MPu is composed of $(\hat{e}, g, g_1, g_2, h, u', m', \vec{u}, \vec{m})$.This master public key is known to every peer in the system and is used for encryption and signature verification. The *Master Private key* MPr is (φ, g_2^α), and is only known to the key server. The master private key is used for generating private keys for publishers and subscribers.

5.2 Key Generation for Publishers/Subscribers

Publisher keys. Before starting to publish events, a publisher contacts the key server along with the credentials for each attribute in its advertisement. If the publisher is allowed to publish events according to its credentials, the key server will generate separate private keys for each credential. Let $Cred_{i,j}$ denote the credential with label j for the attribute A_i, for example, $Cred_{Temp,0}$ denotes credential 0 of attribute $Temp$. The public key of a publisher p for credential $Cred_{i,j}$ is generated as

$$Pu^p_{i,j} := (Cred_{i,j} \parallel A_i \parallel PUB \parallel Epoch).$$

The key server will generate the corresponding private keys as follows: For each credential $Cred_{i,j}$ and a publisher p, let $v_p = H_1(Pu^p_{i,j})$ be a bit string of length n_u and let $v_p[k]$ denote the kth bit. Let $\Gamma_{i,j} \subseteq \{1, 2, \ldots, n_u\}$ be the set of all k for which $v_p[k] = 1$. The key server chooses $\gamma_{i,j} \in \mathbb{Z}_q$ at random and computes

$$Pr^p_{i,j} := \left(g_2^\alpha \left(u' \prod_{k \in \Gamma_{i,j}} u_k \right)^{\gamma_{i,j}}, g^{\gamma_{i,j}} \right) =: \left(Pr^p_{i,j}[1], Pr^p_{i,j}[2] \right).$$

Subscriber keys. Similarly, to receive events matching its subscription, a subscriber should contact the key server and receive the private keys for the credentials associated with each attribute A_i. In case of subscribers, the public key for a credential $Cred_{i,j}$ is given as

$$Pu^s_{i,j} := (Cred_{i,j} \parallel A_i \parallel SUB \parallel Epoch).$$

A different symbol SUB is used to differentiate the keys used for the verification of valid events from the ones used to provide event confidentiality. The private keys are generated as follows: The key server chooses $\gamma_s \in \mathbb{Z}_q$ at random. The same γ_s is used for all credentials associated with a subscription. For each credential $Cred_{i,j}$, it calculates $\Gamma_{i,j}$ similar to the publisher's case, chooses $\gamma_{i,j} \in \mathbb{Z}_q$ and computes

$$\begin{aligned} Pr^s_{i,j} :&= \left(g_2^{\gamma_s} \left(u' \prod_{k \in \Gamma_{i,j}} u_k \right)^{\gamma_{i,j}}, g^{\gamma_{i,j}}, H_3 \left(u' \prod_{k \in \Gamma_{i,j}} u_k \right)^{\varphi} \right) \\ &=: \left(Pr^s_{i,j}[1], Pr^s_{i,j}[2], Pr^s_{i,j}[3] \right). \end{aligned}$$

Furthermore, a credential independent key $Pr^s[4] = g_2^{\frac{\gamma_s + \alpha}{\varphi}}$ is generated. Later, we will see that γ_s along with $Pr^s[4]$ is needed to bind the keys/credentials of a subscription together. It is worth mentioning that the key $Pr^s_{i,j}[3]$ is not used to decrypt events but rather it facilitates the routing of encrypted events from publishers to subscribers (cf. Section 6.4).

5.3 Publishing Events

Encryption. When a publisher wants to publish an event message M, it chooses $b_i \in \mathbb{Z}_q$ at random for each attribute A_i of the event, such that $b = \sum_{i=1}^{d} b_i$. These random values

ensure that only the subscribers who have matching credentials for each of the attributes should be able to decrypt the event. Furthermore, the publisher generates a fixed-length random key SK for each event. More precisely, the following steps are performed by the publisher to encrypt an event:

Step 1. Compute: $CT_1 = \hat{e}(g_1, g_2)^b SK$, $CT_2 = h^b$ and, $CT_3 = \text{BlockCipher}(Msg \| 0^*)^{SK}$, where $Msg = (M, \{Pu^p_{i,j}\})$ defines a record that includes 1) the actual event message M, and 2) the public keys of the credentials which authorize the publisher p to send the event.

The cost of asymmetric encryption generally increases with the size of the plaintext. Therefore, only a fixed-length random key SK is encrypted using the private keys of publisher. The record Msg is encrypted with a symmetric encryption algorithm such as AES [15] or Triple DES [3], using key SK.

During decryption, a subscriber does not know about the credentials with which the event is encrypted and cannot tell in advance whether he is authorized to read the event message. Therefore, to enable the subscribers to detect the successful decryption of events, Msg is appended with a predefined number of zeros $(Msg \| 0^*)$. Alternatively, hash of Msg, i.e., $H_2(Msg)$, can be included in the ciphertext to serve the same purpose.

Step 2. For each attribute A_i, compute $CT_i = g^{b_i}$. The CT_i ciphertexts along with $CT'_{i,j}$ (created in *Step 3*) and $Pr^s_{i,j}[3]$ are used for the routing of encrypted events (cf. Section 6.4).

Step 3. For each attribute of the event, a ciphertext should be created for every credential that matches the value associated with that attribute, so that a subscriber with any of these credentials should be able to decrypt the event. For example, in case of a numeric attribute with value mapped to 0000, a ciphertext should be disseminated for the credentials 0000, 000, 00, and 0.

For each credential $Cred_{i,j}$ that matches the value of the attribute A_i, compute $CT_{i,j} = (u' \prod_{k \in \Gamma_{i,j}} u_k)^{b_i}$ and $CT'_{i,j} = H_4(\hat{e}(H_3(u' \prod_{k \in \Gamma_{i,j}} u_k), h^{b_i}))$, where $\Gamma_{i,j}$ is calculated, as described above. The ciphertexts are ordered according to the containment relationship (in descending order) between their associated credentials, for example, for the above example the order is $[CT_{i,0}, CT_{i,00}, CT_{i,000}, CT_{i,0000}]$.

Signature. Finally, the publisher p signs the ciphertexts using its private keys. It computes $v_m = H_2(M)$ a bit string of length n_m. Let $v_m[k]$ denotes the kth bit and $\Gamma_m \subseteq \{1, 2, \ldots, n_m\}$ be the set of all k for which $v_m[k] = 1$. For each attribute, the credential $Cred_{i,j}$ that authorizes the publisher p to send the corresponding attribute value, p computes

$$CT^{sign}_{i,j}[1] = Pr^p_{i,j}[1]\left(m' \prod_{k \in \Gamma_m} m_k\right)^{b_i},\ CT^{sign}_{i,j}[2] = Pr^p_{i,j}[2].$$

The credentials $Cred_{i,j}$ are same to those included in CT_3.

5.4 Receiving Events

Decryption. On receiving the ciphertexts, a subscriber tries to decrypt them using its private keys. The ciphertexts for each attribute are strictly ordered according to the containment relation between their associated credentials; therefore, a subscriber only tries to decrypt the ciphertext whose position coincides with the position of its credential in the containment hierarchy of the corresponding attribute. The position of a credential can be easily determined by calculating its length. For example, for a numeric attribute, credential 0000 occupies fourth position in the containment hierarchy, i.e., after 0, 00, and 000. Subscribers decrypt the ciphertext in the following manner:

Step 1. The symmetric key SK is retrieved from the ciphertext CT_1 by performing the following pairing-based cryptographic operations:

$$DT = \frac{\left(\prod_{i=1}^{d} \frac{\hat{e}(Pr^s_{i,\tau_i}[1], CT_i)}{\hat{e}(Pr^s_{i,\tau_i}[2], CT_{i,\tau_i})}\right) CT_1}{\hat{e}(CT_2, Pr^s[4])} = SK, \quad (1)$$

where τ_i is the credential assigned to the subscriber for the attribute A_i.[1] As mentioned above, for each attribute A_i, only the ciphertext that corresponds to the credential assigned to the subscriber is used during decryption, i.e., CT_{i,τ_i}. The correctness of the operations performed by (1) is discussed in the supplemental document available online.

Step 2. Symmetric key SK is then used to recover $Msg = (M, \{Pu^p_{i,j}\})$ from CT_3. The successful decryption of Msg is detected by looking for predefined number of zeros appending the Msg record or verifying the hash of Msg, i.e., $H_2(Msg)$.

Verification. A subscriber will only accept the message if it is from an authorized publisher. To check the authenticity of an event, subscribers use the master public key (MPu) and perform the following steps:

Step 1. Compute: $VT_L = \hat{e}(\prod_{i=1}^{d} CT^{sign}_{i,j}[1], g)$, where $\prod_{i=1}^{d} CT^{sign}_{i,j}[1]$ represents the product of all received $CT^{sign}_{i,j}[1]$ ciphertexts.

Step 2. Compute: $VT_{R1} = \prod_{i=1}^{d} \hat{e}(g_1, g_2)$.

Step 3. Compute: $VT_{R2} = \hat{e}(\prod_{i=1}^{d} (u' \prod_{k \in \Gamma_{i,j}} u_k), \prod_{i=1}^{d} CT^{sign}_{i,j}[2])$, where $\prod_{i=1}^{d}(u' \prod_{k \in \Gamma_{i,j}} u_k)$ represents the product of all $Pu^p_{i,j}$ in CT_3 and $\prod_{i=1}^{d} CT^{sign}_{i,j}[2]$ is the product of all received $CT^{sign}_{i,j}[2]$ ciphertexts.

Step 4. Compute: $VT_{R3} = \hat{e}(m' \prod_{k \in \Gamma_m} m_k, \prod_{i=1}^{d} CT_i)$.

The received event is authentic if the following identity holds (see supplemental document, available online, for details): $VT_L = VT_{R1} \times VT_{R2} \times VT_{R3}$.

Table 1 shows the worst case costs of our security methods for numeric and string attributes.

To ensure forward and backward secrecy in the presence of continuously arriving and leaving subscribers, we proposed an epoch-based key management scheme that provides fine-grain access control in a scalable manner. The details are available in the supplemental document available online.

6 Subscription Confidentiality

In this section, we address to achieve subscription confidentiality in a broker-less pub/sub system.

6.1 Publish/Subscribe Overlay

The pub/sub overlay is a virtual forest of logical trees, where each tree is associated with an attribute (cf. Fig. 5). A

1. A subscriber might have many credentials for a single attribute, for example, $\log_2(\mathcal{Z}_i)$ in the worst case for a numeric attribute A_i. Our overlay topology maintenance (cf. Section 6.3) and event dissemination (cf. Section 6.4) mechanisms ensure that a subscriber knows the exact credential needed to decrypt the event.

TABLE 1
Cost of Security Methods

	Public params	Private keys	Ciphertext Size	Encryption cost	Decryption cost	Sign cost	Verification cost
Numeric	$O(1)$	$O\left(\sum_{i=1}^{d} \log_2 \mathcal{Z}_i\right)$	$O\left(\sum_{i=1}^{d} \log_2 \mathcal{Z}_i\right)$	$O\left(\sum_{i=1}^{d} \log_2 \mathcal{Z}_i\right)$	$O(d)$	$O(d)$	$O(d)$
String	$O(1)$	$O\left(\sum_{i=1}^{d} \mathcal{L}_i\right)$	$O\left(\sum_{i=1}^{d} \mathcal{L}_i\right)$	$O\left(\sum_{i=1}^{d} \mathcal{L}_i\right)$	$O(d)$	$O(d)$	$O(d)$

subscriber joins the trees corresponding to the attributes of its subscription. Similarly, a publisher sends an event on all the trees associated with the attributes in the event.

Within each attribute tree, subscribers are connected according to the containment relationship between their credentials associated with the attribute. The subscribers with coarser credentials (e.g., the ones mapped to coarser subspaces in case of numeric attributes) are placed near the root of the tree and forward events to the subscribers with finer credentials. A subscriber with more than one credentials can be handled by running multiple virtual peers on a single physical node, each virtual peer maintaining its own set of tree links, as shown in Fig. 5. To connect to an attribute tree, a newly arriving subscriber s_n sends the connection request along with its credential to a random peer s_r in the tree. The peer s_r compares the request credential with its own; if the peer's credential covers the request credential and the peer can accommodate more children, it accepts the connection. Otherwise, the connection request is forwarded to all the children with covering credentials and the parent peer with the exception of the peer from which it was received. In this way, the connection request is forwarded by many peers in the tree before it reaches the suitable peer with covering credential and available connection, as shown in Fig. 5.

6.2 Weak Subscription Confidentiality

It is infeasible to provide strong subscription confidentiality in a broker-less pub/sub system because the maintenance of the overlay topology requires each peer to know the subscription of its parent as well as its children. To address this issue, a weaker notion of subscription confidentiality is required.

Definition 6.1. *Let s_1 and s_2 denote two subscribers in a pub/sub system which both possess credentials for an attribute A_i. Weak subscription confidentiality ensures that at most the following information can be inferred about the credentials of the subscribers:*

1. *The credential of s_1 is either coarser or equal to the credentials of s_2.*
2. *The credential of s_1 is either finer or equal to the credentials of s_2.*
3. *The credentials of s_1 and s_2 are not in any containment relationship.*

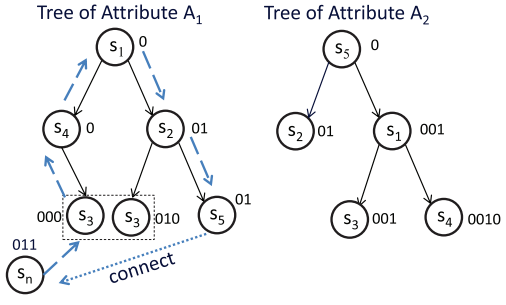


Fig. 5. Pub/Sub system with two numeric attributes.

6.3 Secure Overlay Maintenance

In the following, we propose a secure protocol to maintain the desired pub/sub overlay topology without violating the weak subscription confidentiality. For simplicity and without loss of generality, here we discuss the overlay maintenance w.r.t. a single tree associated with a numeric attribute A_i and each of the subscribers owns a single credential.

The secure overlay maintenance protocol is based on the idea that in the tree, subscribers are always connected according to the containment relationship between their credentials, for example, a subscriber with credential 00 can only connect to the subscribers with credentials 0 or 00.

A new subscriber s generates a random key SW and encrypts it with the public keys $Pu^s_{i,j}$ for all credentials that cover its own credential, for example, a subscriber with credential 00 will generate ciphertexts by applying the public keys $Pu^s_{i,0}$ and $Pu^s_{i,00}$. The generated ciphertexts are added to a *connection request* (CR) and the request is forwarded to a random peer in the tree. A connection is established if the peer can decrypt any of the ciphertexts using its private keys.

Filling the security gaps. By looking at the number of ciphertexts in the connection request, a peer can detect the credential of the requesting subscriber s. For example, a subscriber with credential 00 can only connect to 0 or 00, and therefore, a connection request will have two ciphertexts, whereas the connection request for 000 will have three ciphertexts. In the worst case, a subscriber has a credential of the finest granularity. This can be covered by $\log_2(\mathcal{Z}_i)$ other credentials, and therefore, a connection request contains in the worst case that many ciphertexts. To avoid any information leak, ciphertexts in the connection request are always kept in $O(\log_2 \mathcal{Z}_i)$ ($O(\mathcal{L}_i)$ for prefix matching) by adding random ciphertexts if needed. Furthermore, the ciphertexts are shuffled to avoid any information leak from their order.

A different random key SW is used for the generation of each ciphertext to avoid any information leak to the peer who has successfully decrypted one of the ciphertexts and, thus, has recovered the random key SW. Otherwise, the peer can try to generate ciphertexts by encrypting the (recovered) SW with public keys for $O(\log_2 \mathcal{Z}_i)$ (likewise $O(\mathcal{L}_i)$) credentials and can easily determine the random ciphertexts in the connection request and, thus, the

credentials of the requesting subscriber s. Finally, to avoid an attacker to generate arbitrary connection request messages and try to discover the credential of other peers in the system, the connection request is signed by the key server. This step needs to be performed only once, when a newly arriving subscriber authorizes itself to the key server in order to receive private keys for its credentials.

Overall algorithm. The secure overlay maintenance protocol is shown in Algorithm 1. In the algorithm, the procedure *decrypt_request* tries to decrypt one of the ciphertexts in the connection request message.

Algorithm 1. Secure overlay maintenance protocol at peer s_q.

```
1:  upon event Receive(CR of s_new from s_p) do
2:    if decrypt_request(CR) == SUCCESS then
3:      if degree(s_q) == available then //can have
        child peers
4:          connect to the s_new
5:      else
6:          forward CR to {child peers and parent} − s_p
7:    if decrypt_request(CR) == FAIL then
8:      if s_p == parent then
9:          Try to swap by sending its own CR to the s_new.
10:     else
11:         forward to parent
```

A child peer s_q receives CR (of subscriber s_{new}) from the parent s_p only if the parent cannot accommodate more children. If s_q cannot be the parent of s_{new}, i.e., s_{new}'s credential is coarser than that of s_q, then it tries to swap its position with s_{new} by sending its own connection request (cf. *Algorithm* 1, *lines 7-9*). However, if none of the children of parent s_p can connect or swap with s_{new}, then there is no containment relationship between the credentials of the children and s_{new}. In this case, a parent should disconnect one of its children to ensure the new subscriber is connected to the tree.

6.4 Secure Event Dissemination

To publish an event, a publisher forwards the ciphertexts of each attribute to the root of the corresponding attribute tree. All the ciphertexts of an event are labeled with a unique value such as sequence number of the event. This helps subscribers to identify all the ciphertexts of an event (though the ciphertexts for each attribute are received on a separate tree). In this section, we describe two strategies to route events (from publishers to the relevant subscribers) in the pub/sub overlay network without violating the weak subscription confidentiality.

One-hop flooding (OHF). In one-hop flooding, a parent assumes that the children have the same credentials as its own and forward each successfully decrypted event to all of them. In turn, the children forward each event which was successfully decrypted to all of their children and so on. In this strategy, each subscriber maintains $O(d \log_2 \mathcal{Z})$ overlay connections in worst case (one for each credential). Moreover, a child may have finer credentials than its parent and may receive false positives. This strategy is detailed in the supplemental document available online.

Multicredential routing (MCR). MCR strategy targets reduction in false positives by enabling parents to forward only those event on each attribute tree that match the credential of their children. In particular, every child subscriber s on an attribute tree A_i informs each parent p about the private key $Pr^s_{i,\tau_i}[3]$ of the credential Cred_{i,τ_i} associated with the overlay connection to p. Upon receiving an event on an attribute tree A_i, a parent p forwards the event to a child s if one of the credentials (i.e., $CT'_{i,j}$) under which the event is encrypted matches the credential of the private key $Pr^s_{i,j}[3]$ submitted by the child s. More precisely, the decision (DEC) to forward ciphertexts associated with an attribute A_i to the child can be described as

$$\mathrm{DEC} = \begin{cases} \text{forward}, & \text{if } H_4(\hat{e}(Pr^s_{i,\tau_i}[3], CT_i)) = CT'_{i,\tau_i}, \\ \text{drop}, & \text{otherwise.} \end{cases} \quad (2)$$

Although the actual credentials of children are hidden from the parent peers by the use of $Pr^s_{i,j}[3]$ keys. Nevertheless, the hidden credentials (i.e., $Pr^s_{i,j}[3]$ keys) are not adequate to ensure weak subscription confidentiality. This is because a parent decrypts every event which it forward to its children and, therefore, can eventually discover their credentials, for example, by maintaining histories of the events forwarded to each child.

To preserve weak subscription confidentiality, subscribers divide the original credential(s) for each attribute of their subscriptions into a number of fine granular credentials and $Pr^s_{i,j}[3]$ key for each (fine granular) credential is forwarded to a separate parent in the corresponding attribute tree. For example, credential 1 for a numeric attribute can be divided into three credentials 10, 110, and 111, and a separate parent connection can be maintained (by forwarding $Pr^s_{i,j}[3]$ key) for each credential (obtained as a result of division). This enables that the exact credential(s) of an attribute of a subscription cannot be determined unless multiple parents (with knowledge about the individual credentials) collude with each other. To ensure that a subscriber always connects to a distinct parent for each of its credentials, techniques such as broadcast revocation can be used [13]. It is also important to mention that the subscribers cannot generate $Pr^s_{i,j}[3]$ keys for the fine granular credentials obtained as a result of dividing the original credential(s) and, therefore, should contact the key server for the creation of $Pr^s_{i,j}[3]$ keys. This step can be performed at the same time when a new subscriber authorizes itself to the key server.

A subscriber maintains at least $k_{\mathcal{D}}$ credentials for each attribute of its subscription. The parameter $k_{\mathcal{D}}$ can be defined by the system or selected by each subscriber independently depending on its confidentiality requirements. In this strategy, a subscriber may maintain more overlay connections than OHF if the number of credentials per attribute (i.e., $k_{\mathcal{D}}$) is more than $\log_2 \mathcal{Z}$. However, the complete subscription of the subscriber cannot be determined unless $d.k_{\mathcal{D}}$ parents collude with each other.

Analysis of secure overlay maintenance (cf. Section 6.2) and secure event dissemination (cf. Section 6.4) algorithms to preserve weaker notion of subscription confidentiality as well as traffic analysis and timing attacks on subscription

TABLE 2
Throughput of Cryptographic Primitives

Encryption(E)	10KB/sec
Decryption(D)	10KB/sec
Signature(S)	158 sign/sec
Verification(V)	52 verify/sec

confidentiality are discussed in the supplemental document available online.

7 Performance Evaluations

We evaluate three aspects of our system: 1) quantifying the overhead of our cryptographic primitives, 2) benchmarking the performance of our secure pub/sub system, and 3) analyzing attacks on subscription confidentiality. Here, we only discuss the first two aspects and the evaluations related to the analysis of subscription confidentiality are available in the supplemental document available online.

Experimental Setup. Simulations are performed using PeerSim [11]. Simulations are performed for up to $N = 2,048$ peers. Unless otherwise stated, out-degree constraints of the peers are chosen as $\log_2(N)$. The delays between the communication links are chosen in the range [24 and 134 ms]. The complex subscriptions used during the evaluations contain conjunction of predicates defined on up to $d = 16$ different attributes. We evaluate the system performance under uniform (WL_1) and skewed (WL_2) subscription workloads, and with a uniform and skewed event distribution. Skew is simulated using the widely used 80-20 percent Zipfian distribution with three to five hot spots. The security mechanisms are implemented by the pairing-based cryptography library [14]. The implementation uses a 160-bit elliptic curve group based on the supersingular curve $y^2 = x^3 + x$ over a 512-bit finite field.

7.1 Performance of Cryptographic Primitives

In this section, we measure the computational overhead of our security methods. All of our measurements were made on a 2-GHz Intel Centrino Duo with 2-GB RAM, running Ubuntu 9. Table 2 shows the throughput of the cryptographic primitives to perform encryption, decryption, signature, and verification. All reporting values are averaged over 1,000 measurements. In our system, pairing-based encryption is used to encrypt a random key SK, which is later used to decrypt the actual event using symmetric encryption (cf. Section 5.3). Therefore, the message size is kept 128 bytes as this key length is good enough for most symmetric encryption algorithms. Table 3 shows the computational overhead (in msec) from the perspective of publishers and subscribers in our system. In general, the cost of verification is high due to the fact that it involves the computationally expensive pairing operations. Likewise, Table 4 shows the average CPU utilization for publishers and subscribers.

TABLE 3
Computation Times for Publishers and Subscribers

Operation	Time(msec)
Encryption(E)	$6.9 + d \times 5.4$
Signature(S)	$d \times 6.32$
Decryption(D)	$6.2 + d \times 6.1$
Verification(V)	$19.3 + d \times 0.001$

TABLE 4
Average CPU Utilization

Operation	Usage (%)
Encryption(E)	$0.3 + d \times 0.24$
Signature(S)	$d \times 0.274$
Decryption(D)	$0.266 + d \times 0.265$
Verification(V)	$0.83 + d \times 0.00003$

7.2 Performance of Publish/Subscribe System

The pub/sub overlay proposed in Section 6.1 is similar to DPS system [1] with modifications to ensure subscription confidentiality. In this paper, we, therefore, evaluate performance and scalability of the proposed pub/sub system only with respect to the security mechanisms and omit other aspects. In particular, we evaluate the performance of our system w.r.t. the overlay construction time and the event dissemination delays.

In Fig. 6a, we measure the average delay experienced by each subscriber to connect to a suitable position in an attribute tree. Delay is measured from the time a subscriber sends connection request message to a random peer in the tree till the time the connection is actually established. The evaluations are performed only for a single attribute tree. Fig. 6a shows that the average connection time (delay) increases with the number of peers in the system because of the increase in the height of the attribute tree (each new hop increases the network delay as well as time to apply security methods). Furthermore, Fig. 6a shows that there is

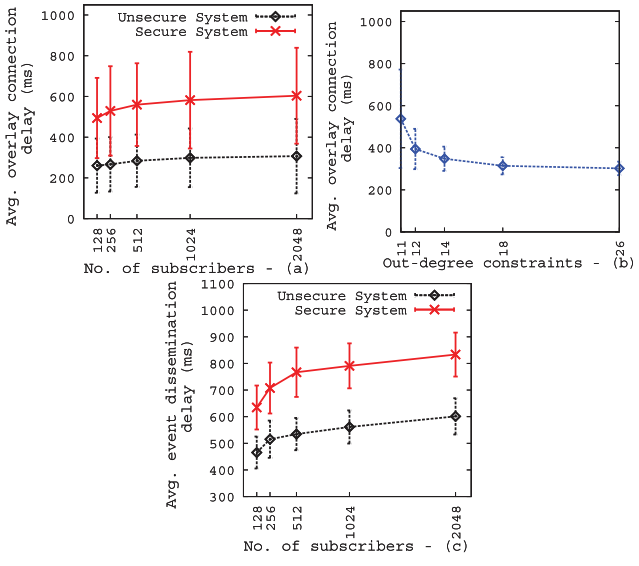


Fig. 6. Performance evaluations.

an overhead of approximately 230-300 ms due to security mechanisms. Our evaluations with higher number of attributes indicate that the average connection delay experienced by a subscriber is independent to the number of attributes. The reason being that each attribute tree is created in parallel and a subscriber sends connection request to connect multiple attribute trees at the same time. Nevertheless, the average connection delay is affected by the out-degree constraints of the peers participating in the overlay network. Fig. 6b shows that for a fixed set of peers (i.e., 1,024 peers for this experiment), the average delay experienced by subscribers decreases significantly with the increase in out-degree mainly because the resultant dissemination tree is fat (i.e., tree with smaller height). For the similar reason, Fig. 6a reports that average connection time (delay) increases very slightly with the number of peers. The increase in the average connection delay is small because the overall out-degree also increases with the number of peers, resulting in only a small increase in the height of the tree.

Fig. 6c measures the average time needed by the event to be disseminated to all the relevant subscribers in the system. For each subscriber, the time is measured from the dissemination of the event by the publisher till it is successfully decrypted and verified by the subscriber. For the experiment, 160 publishers are introduced in the system and each published 10 events. Fig. 6c shows that the average time to disseminate an event increases with the number of peers in the system because of the increase in number of the relevant subscribers as well as the height of the dissemination tree. Similar to the previous results, there is an overhead of approximately 150-250 ms due to security mechanisms.

The evaluation results obtained from WL_2 show similar trend. The rest of the evaluations can be found in the supplemental document available online.

8 Conclusion

In this paper, we have presented a new approach to provide authentication and confidentiality in a broker-less content-based pub/sub system. The approach is highly scalable in terms of number of subscribers and publishers in the system and the number of keys maintained by them. In particular, we have developed mechanisms to assign credentials to publishers and subscribers according to their subscriptions and advertisements. Private keys assigned to publishers and subscribers, and the ciphertexts are labeled with credentials. We adapted techniques from identity-based encryption 1) to ensure that a particular subscriber can decrypt an event only if there is a match between the credentials associated with the event and its private keys and 2) to allow subscribers to verify the authenticity of received events. Furthermore, we developed a secure overlay maintenance protocol and proposed two event dissemination strategies to preserve the weak subscription confidentiality in the presence of semantic clustering of subscribers. The evaluations demonstrate the viability of the proposed security mechanisms and analyze attacks on subscription confidentiality.

References

[1] E. Anceaume, M. Gradinariu, A.K. Datta, G. Simon, and A. Virgillito, "A Semantic Overlay for Self- Peer-to-Peer Publish/Subscribe," *Proc. 26th IEEE Int'l Conf. Distributed Computing Systems (ICDCS)*, 2006.

[2] J. Bacon, D.M. Eyers, J. Singh, and P.R. Pietzuch, "Access Control in Publish/Subscribe Systems," *Proc. Second ACM Int'l Conf. Distributed Event-Based Systems (DEBS)*, 2008.

[3] W.C. Barker and E.B. Barker, "SP 800-67 Rev. 1. Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher," technical report, Nat'l Inst. of Standards & Technology, 2012.

[4] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-Policy Attribute-Based Encryption," *Proc. IEEE Symp. Security and Privacy*, 2007.

[5] D. Boneh, G.D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public Key Encryption with Keyword Search," *Proc. Int'l Conf. Theory and Applications of Cryptographic Techniques on Advances in Cryptology (EUROCRYPT)*, 2004.

[6] D. Boneh and M.K. Franklin, "Identity-Based Encryption from the Weil Pairing," *Proc. Int'l Cryptology Conf. Advances in Cryptology*, 2001.

[7] S. Choi, G. Ghinita, and E. Bertino, "A Privacy-Enhancing Content-Based Publish/Subscribe System Using Scalar Product Preserving Transformations," *Proc. 21st Int'l Conf. Database and Expert Systems Applications: Part I*, 2010.

[8] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data," *Proc. ACM 13th Conf. Computer and Comm. Security (CCS)*, 2006.

[9] M. Ion, G. Russello, and B. Crispo, "Supporting Publication and Subscription Confidentiality in Pub/Sub Networks," *Proc. Sixth Int'l ICST Conf. Security and Privacy in Comm. Networks (SecureComm)*, 2010.

[10] H.-A. Jacobsen, A.K.Y. Cheung, G. Li, B. Maniymaran, V. Muthusamy, and R.S. Kazemzadeh, "The PADRES Publish/Subscribe System," *Principles and Applications of Distributed Event-Based Systems*. IGI Global, 2010.

[11] M. Jelasity, A. Montresor, G.P. Jesi, and S. Voulgaris, "PeerSim: A Peer-to-Peer Simulator," http://peersim.sourceforge.net/, 2013.

[12] H. Khurana, "Scalable Security and Accounting Services for Content-Based Publish/Subscribe Systems," *Proc. ACM Symp. Applied Computing*, 2005.

[13] A. Lewko, A. Sahai, and B. Waters, "Revocation Systems with Very Small Private Keys," *Proc. IEEE Symp. Security and Privacy*, 2010.

[14] B. Lynn, "The Pairing-Based Cryptography (PBC) Library," http://crypto.stanford.edu/pbc/, 2010.

[15] F.P. Miller, A.F. Vandome, and J. McBrewster, *Advanced Encryption Standard*. Alpha Press, 2009.

[16] M. Nabeel, N. Shang, and E. Bertino, "Efficient Privacy Preserving Content Based Publish Subscribe Systems," *Proc. 17th ACM Symp. Access Control Models and Technologies*, 2012.

[17] L. Opyrchal and A. Prakash, "Secure Distribution of Events in Content-Based Publish Subscribe Systems," *Proc. 10th Conf. USENIX Security Symp.*, 2001.

[18] L.I.W. Pesonen, D.M. Eyers, and J. Bacon, "Encryption-Enforced Access Control in Dynamic Multi-Domain Publish/Subscribe Networks," *Proc. ACM Int'l Conf. Distributed Event-Based Systems (DEBS)*, 2007.

[19] P. Pietzuch, "Hermes: A Scalable Event-Based Middleware," PhD dissertation, Univ. of Cambridge, Feb. 2004.

[20] C. Raiciu and D.S. Rosenblum, "Enabling Confidentiality in Content-Based Publish/Subscribe Infrastructures," *Proc. IEEE Second CreatNet Int'l Conf. Security and Privacy in Comm. Networks (SecureComm)*, 2006.

[21] A. Shikfa, M. Önen, and R. Molva, "Privacy-Preserving Content-Based Publish/Subscribe Networks," *Proc. Emerging Challenges for Security, Privacy and Trust*, 2009.

[22] M. Srivatsa, L. Liu, and A. Iyengar, "EventGuard: A System Architecture for Securing Publish-Subscribe Networks," *ACM Trans. Computer Systems*, vol. 29, article 10, 2011.

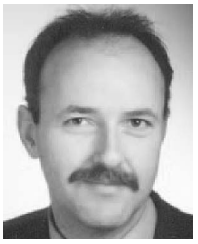
[23] M.A. Tariq, B. Koldehofe, A. Altaweel, and K. Rothermel, "Providing Basic Security Mechanisms in Broker-Less Publish/Subscribe Systems," *Proc. ACM Fourth Int'l Conf. Distributed Event-Based Systems (DEBS)*, 2010.

[24] M.A. Tariq, B. Koldehofe, G.G. Koch, I. Khan, and K. Rothermel, "Meeting Subscriber-Defined QoS Constraints in Publish/Subscribe Systems," *Concurrency and Computation: Practice and Experience*, vol. 23, pp. 2140-2153, 2011.

[25] Y. Yu, B. Yang, Y. Sun, and S.-l. Zhu, "Identity Based Signcryption Scheme without Random Oracles," *Computer Standards & Interfaces*, vol. 31, pp. 56-62, 2009.

Muhammad Adnan Tariq received the PhD degree from the University of Stuttgart, Germany. He is currently working as a postdoctoral research fellow at the Distributed Systems research group, University of Stuttgart. His research interests include event-based systems, service-aware adaptive overlay networks, QoS, and security.

Boris Koldehofe received the PhD degree from the Chalmers University of Technology, in 2005, and joined the EPFL as a postdoctoral research fellow directly after the PhD. He has been a senior researcher and lecturer at the IPVS of the University of Stuttgart in the field of distributed systems, since 2006. In 2010, he was also appointed as a visiting professor at the University of Heidelberg. Currently, he is leading the Adaptive Communication Systems group at the Department of Distributed Systems, University of Stuttgart, where his research is centered on scalable and reliable adaptation of distributed applications. In particular, his current work deals with methods for reliable, mobile, and secure event processing systems and event routing with QoS in communication networks.



Kurt Rothermel received the doctoral degree in computer science from the University of Stuttgart in 1985. From 1986 to 1987, he was a postdoctoral fellow at the IBM Almaden Research Center in San Jose and then joined IBM European Networking Center in Heidelberg, Germany. He left IBM in 1990 to become a professor of computer science back at the University of Stuttgart, where he now leads the distributed systems research group. His research interests are in the fields on distributed systems, computer networks, mobile systems, and sensor networks.