Privacy Preserving Context Aware Publish Subscribe Systems

Mohamed Nabeel[1], Stefan Appel[2] Elisa Bertino[1], and Alejandro Buchmann[2]

[1] Purdue University, West Lafayette, IN, USA,
{nabeel, bertino}@cs.purdue.edu
[2] TU Darmstadt, Darmstadt, Germany,
{appel, buchmann}@dvs.tu-darmstadt.de

Abstract. Publish/subscribe (pub/sub) systems support highly scalable, many-to-many communications among loosely coupled publishers and subscribers. Modern pub/sub systems perform message routing based on the message *content* and allow subscribers to receive messages related to their subscriptions and the current *context*. However, both content and context encode sensitive information which should be protected from third-party brokers that make routing decisions. In this work, we address this issue by proposing an approach for constructing a privacy preserving context-based pub/sub system. In particular, our approach assures the confidentiality of the messages being published and subscriptions being issued while allowing the brokers to make routing decisions without decrypting individual messages and subscriptions, and without learning the context. Further, subscribers with a frequently changing context such as location are able to issue and update subscriptions without revealing the subscriptions in plaintext to the broker and without the need to contact a trusted third party for each subscription change resulting from a change in the context. Our approach is based on a modified version of the Paillier additive homomorphic cryptosystem and a recent expressive group key management scheme. The former construct is used to perform privacy preserving matching and covering, and the latter construct is used to enforce fine-grained encryption based access control on the messages being published. We optimize our approach in order to efficiently handle frequently changing contexts. We have implemented our approach in a prototype using an industry strength JMS broker middleware. The experimental results show that our approach is highly practical.

1 Introduction

The publish/subscribe (pub/sub) paradigm is a well known approach for disseminating information between multiple interested parties in a decoupled and asynchronous manner [9]. Message producers submit messages to a broker network which routes those to interested subscribers. Subscribers express their interest in specific messages by issuing subscriptions. Content-based pub/sub systems allow subscribers to express their interest based upon the message content. This content can be an arbitrary payload, e.g., a set of attribute-value (att/val) pairs, XML documents, or combinations of different types. The supported message content depends on the pub/sub middleware which performs message routing and matching. In many systems it is common to use att/val pairs

to describe content and to express subscription filters as logical expressions on these attributes (e.g., MessageType == StockTickMessage $\wedge$ StockPrice > 38).

Context-sensitive message dissemination extends pub/sub content dissemination by taking into account the subscriber context [8]. Subscribers express interest in messages based on their current context, e.g., their current location. A major challenge for context-sensitive message dissemination is that the context of subscribers, and thus the subscriptions, change frequently over time, e.g., as the location changes. Context-dependent information dissemination is however a crucial requirement in many application scenarios. One example is a traffic information system (TIS) where information about the traffic situation is provided to interested parties. Due to the characteristics of this scenario a pub/sub middleware supporting context-sensitive message dissemination is an appropriate infrastructure. Information about the current traffic situation is published and participants express their interest in information with subscriptions. As subscribers move and thus their contexts constantly change, subscriptions need to change accordingly, e.g., subscribers would typically be interested in traffic information along the route they are traveling.

A major shortcoming of existing context-based pub/sub approaches is that they do not assure privacy during message dissemination. In such systems, the broker receives the subscriptions in plaintext and is thus aware of the context of subscribers. In case of the TIS example, this implies that brokers are aware of the exact position of subscribers. In order to assure privacy, we propose an approach to construct a privacy-preserving context-based pub/sub system. We extend and improve the idea presented in our preliminary work [17] in order to propose a new security model and construct our privacy preserving context-based pub/sub system.

In the previous model [17], each subscriber is required to submit a new subscription via a secure channel to a trusted third-party (TTP) that encrypts the subscription in a special way. This special encryption operation is called *blinding* and the encrypted value *blinded value*. Such blinded values are semantically secure (IND-CPA secure) where two blindings of the same value result in two different blinded values. The subscriber then registers with such blinded subscriptions at an untrusted broker. Such an approach allows the use of honest-but-curious brokers[3]to perform matching and routing on encrypted notifications using the blinded subscriptions. Although such an approach is privacy preserving, it is not suitable for context-based pub/sub systems. The reason is that as the context of subscribers changes frequently, subscriptions have to be updated often and involving a TTP to blind every subscription is not any longer feasible. Further, the previous model does not support fine-grained access control of notifications. Thus a new security model and mechanisms are required whereby subscribers are allowed to create their own blinded subscriptions without compromising the security of the overall system and enforce fine-grained access control of notifications. In this work, we achieve our first objective by allowing authorized subscribers to create blinded subscriptions after obtaining some public security parameters at the time of registration. After the initial interactions, subscribers are not required to contact a TTP unless the public security parameters are updated. We achieve the second objective by introducing

[3] Brokers are obliged to follow the protocol, but they are curious to learn as much as possible during the execution of the protocol.

a fine-grained encryption-based access control mechanism. An advantage of such an approach compared to approaches based on shared secrets is that no secret information is given to subscribers to generate blinded subscriptions and therefore our approach avoids the problem of leakage of shared secrets by malicious subscribers.

Each notification in our approach is encrypted twice. The first encryption, referred to as blinding operation, is performed to blind each attribute value in the notification that is used by brokers to perform matching operations. The notification blinding is similar to the subscription blinding operation mentioned earlier except that the two operations use different blinding parameters so that certain parameters cancel off when a blinded notification and a subscription are homomorphically added by multiplying them. It should be noted that brokers cannot decrypt individual blinded values and they only learn a randomized difference between subscription and notification values when they perform matching operations. The second encryption, referred to as broadcast encryption, is performed to encrypt the payload of notifications based on fine-grained access control policies (ACPs). Inline with the current initiatives on identity management [21], fine-grained ACPs are specified using the attributes of subscribers, referred to as *identity attributes*. Our broadcast encryption is based on a recently proposed group key management (GKM) scheme, referred to as attribute based GKM (AB-GKM) [24, 19, 20]. In the AB-GKM scheme, unlike conventional GKM schemes [3, 13], subscribers are allowed to dynamically derive the data decryption keys based on the attribute credentials they possess and some public information provided by publishers.

We also provide support for multiple publishers that produce messages with overlapping attribute sets; a blinded subscription may match notifications from several publishers. Especially for context-based pub/sub, it is a crucial requirement to support multiple publishers located in the same context, e.g., the same geographical region. We thus introduce context managers as TTPs in our approach. Context managers provide publishers as well as subscribers with information required to publish encrypted/blinded notifications and to issue blinded subscriptions. Once publishers and subscribers obtain the required security parameters, the context manager is responsible for controlling the level of protection. It decides when to renew and redistribute security parameters to publishers and subscriber in order to reduce the risk of adversaries learning the content of notifications and subscriptions.

Figure 1 shows a block diagram of our pub/sub system. Publishers and subscribers belonging to different contexts are connected to the pub/sub system for message exchange. The context managers are contacted prior to establishing a connection to the pub/sub system to obtain the security parameters and secrets required for the encryption and blinding process.

We implement our scheme based on the Java Message Service (JMS), the de-facto industry standard for messaging. We chose Apache ActiveMQ as JMS broker and extended it to support subscription evaluation on encrypted data. This allows us to perform a realistic evaluation of our approach since ActiveMQ is used in many real-world production pub/sub systems.

Our paper is structured as follows. Section 2 introduces the cryptographic constructs and the group key management scheme used in our approach. Section 3 introduces context-based pub/sub systems and presents an overview of our solution. Sections 4

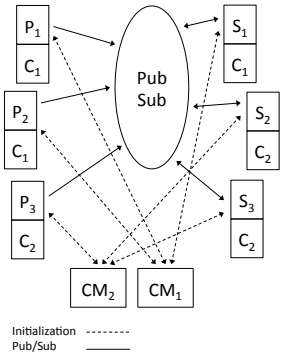


Fig. 1. System Overview (P: Publisher, S: Subscriber, C: Context, CM: Context Manager)

and 5 present the technical details of the modified Paillier cryptosystem which is used to blind subscriptions and notifications, and our overall scheme. Section 6 shows experimental results for various algorithms and the overall system implemented in Apache ActiveMQ. Section 7 discusses related work and Section 8 concludes the paper.

2 Background

2.1 Paillier homomorphic cryptosystem

The *Paillier homomorphic cryptosystem* is a public key cryptosystem by Paillier [22] based on the "Composite Residuosity assumption (CRA)". The Paillier cryptosystem is homomorphic in that, by using a public key, the encryption of the sum $m_1 + m_2$ of two messages m_1 and m_2 can be computed from the encryption of m_1 and m_2. Our approach and protocols are inspired by how the Paillier cryptosystem works. Hence, we provide some internal details of the cryptosystem below so that readers can follow the rest of the paper.

Key generation

Set $n = pq$, where p and q are two large prime numbers. Set $\lambda = \text{lcm}(p-1, q-1)$, i.e., the least common multiple of $p-1$ and $q-1$. Randomly select a base $g \in \mathbb{Z}/(n^2)^\times$ such that the order of g_p is a multiple of n. Such a g_p can be efficiently found by randomly choosing $g_p \in \mathbb{Z}/(n^2)^\times$, then verifying that

$$\gcd(L(g_p^\lambda \pmod{n^2}), n)) = 1, \text{ where } L(u) = (u-1)/n \tag{1}$$

for $u \in S_n = \{u < n^2 | u = 1 \pmod{n}\}$. In this case, set $\mu = \left(L(g_p^\lambda \pmod{n^2})\right)^{-1} \pmod{n}$. The public encryption key is a pair (n, g_p). The private decryption key is (λ, μ), or equivalently (p, q, μ).

Encryption $E(m, r)$

Given plaintext $m \in \{0, 1, \ldots, n-1\}$, select a random $r \in \{1, 2, \ldots, n-1\}$, and encrypt m as $E(m, r) = g_p^m \cdot r^n \pmod{n^2}$. When the value of r is not important to

the context, we sometimes simply write a short-hand $E(m)$ instead of $E(m, r)$ for the Paillier ciphertext of m.
Decryption $D(c)$
Given ciphertext $c \in \mathbb{Z}/(n^2)^\times$, decrypt c as

$$D(c) = L(c^\lambda \pmod{n^2}) \cdot \mu \pmod{n}. \tag{2}$$

In the construction of our pub/sub system, the Paillier homomorphic cryptosystem is used in a way that public and private keys are judiciously distributed among publishers, subscribers, and brokers such that the confidentiality and privacy are assured based on homomorphic encryption. A detailed description of the construction is presented in Section 4.

2.2 Attribute Based Group Key Management

Broadcast Group Key Management (BGKM) schemes [6, 1, 29, 24] are a special type of GKM scheme whereby the rekey operation is performed with a single broadcast without requiring private communication channels. Unlike conventional GKM schemes, BGKM schemes do not give subscribers private keys. Instead subscribers are given a secret which is combined with public information to obtain the actual private keys. Such schemes have the advantage of requiring a private communication only once for the initial secret sharing. The subsequent rekeying operations are performed using one broadcast message. Further, in such schemes achieving forward and backward security requires only to change the public information and does not affect the secrets given to existing subscribers. However, BGKM schemes do not support group membership policies over a set of attributes. In their basic form, they can only support 1-*out-of-*n threshold policies by which a group member possessing 1 attribute out of the possible n attributes is able to derive the group key. A recently proposed attribute based GKM (AB-GKM) scheme [18] provides all the benefits of BGKM schemes and also supports attribute based access control policies (ACPs).

Subscribers are required to show their identity attributes to the group controller to obtain secrets using the AB-GKM scheme. In order to hide the identity attributes from the group controller while allowing only valid subscribers to obtain secrets, we utilize oblivious commitment based envelope (OCBE) protocols [12]. We omit the details of the OCBE protocols due to the page limit.

Before presenting the abstract details of the AB-GKM operations, we first introduce a few terms that are useful for describing the operations. Attribute conditions and access control policies are formally defined as follows.

Definition 1 (Attribute Condition). *An attribute condition* cond *is an expression of the form: "*name_{attr} op l*", where* name_{attr} *is the name of an identity attribute* $attr$*,* op *is a comparison operator such as* $=, <, >, \leq, \geq, \neq$*, and* l *is a value that can be assumed by the attribute* $attr$*.*

Definition 2 (Access control policy). *An access control policy ACP is a tuple* (s, o) *where: o denotes a set of notifications,* $\{d_1, \ldots, d_t\}$*;*

and s is a monotonic expression[4] *over a set of attribute conditions that must be satisfied by a subscriber to have access to o.*

We denote the set of all attribute conditions as $\mathcal{ACB}$ and the set of all ACPs as $\mathcal{ACPB}$. Example 1 shows a sample ACP.

Example 1. The ACP ((“type = regular” $\wedge$ “region = Indiana”) $\vee$ “type = premium”, {new movie}) states that a subscriber, either having a premium subscription or having a regular subscription in Indiana region, has access to new movies.

The idea behind the AB-GKM scheme is as follows. A separate BGKM instance for each attribute condition is constructed. The ACP is embedded in an access structure $\mathcal{T}$. $\mathcal{T}$ is a tree with the internal nodes representing threshold gates and the leaves representing BGKM instances for the attributes. $\mathcal{T}$ can represent any monotonic policy. The goal of the access tree is to allow deriving the group key for only the subscribers whose attributes satisfy the access structure $\mathcal{T}$.

The AB-GKM scheme consists of five algorithms: **Setup**, **SecGen**, **KeyGen**, **KeyDer** and **ReKey**. **Setup** initializes the system. **SecGen** generates a unique secret for each attribute condition. For a given ACP, **KeyGen** creates a symmetric key, public information and an access structure. **KeyDer** derives the symmetric key given one or more secrets and public information. **ReKey** regenerates the symmetric key and public information.

3 Overview

As mentioned in Section 1, our privacy preserving context-based pub/sub system is designed as an extension to current pub/sub systems. It requires a modification of the matching algorithm inside the message broker to support the evaluation of blinded subscriptions against blinded notifications without decrypting them. Our system also supports fine-grained encryption based access control over notifications. Publishers encrypt notifications so that only authorized subscribers can derive the key and decrypt the notifications. In order to assure privacy, publishers and subscribers must perform an initialization to obtain secrets and public parameters that they later need for encryption and blinding operations.

In this section, we give an overview of the modified Paillier cryptosystem, which we use to provide privacy preserving matching at brokers, and our context-based pub/sub model and present our system architecture. We describe the initialization phase as well as the regular runtime behavior. Finally, we present the trust model assumed in our approach.

3.1 Modified Paillier Cryptosystem

In our work, we adapt the Paillier cryptosystem so that brokers can perform matching operations without decrypting individual subscriptions and notifications. A high level

[4] Monotonic expressions are Boolean formulas that contain only conjunction and disjunction connectives, but no negation.

overview of the modifications we perform to the Paillier cryptosystem and the rationale behind our modifications are provided below.

We first shift the computation towards encryption so that decryption is computationally more efficient than the Paillier decryption. We also allow brokers to perform certain operations without knowing the private key. Such shifting of computation improves the performance of the overall pub/sub system since publishers and subscribers, which perform encryption, are typically distributed to many nodes while brokers have to handle notifications from many publishers and subscribers. Thus, by making decryption efficient, we eliminate a bottleneck in the system and improve the overall efficiency. We also blind the encrypted values and make μ, a parameter of the Paillier cryptosystem, public so that individual values cannot be decrypted, but a blinded subscription and a blinded notification can be multiplied together to obtain the difference. In order to make the correct matching decision by calculating the difference, we limit the domain size (l bits) of the subscription and notification values. We assume that l is much smaller compared to the plaintext space of the Paillier cryptosystem, n. For example, the subscription and notification for the attribute age, which may take values from 0 to 200, can be represented using a domain size of 8 bits. Since the domain size is much smaller than the plaintext space, brokers can make the matching decision by calculating the difference as follows: If the difference between a notification and a subscription value is in the first half of the plaintext space, the difference is positive and the notification value is greater than or equal to the subscription value. Otherwise, the difference is negative and the notification value is less than the subscription value.

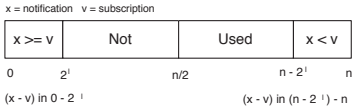


Fig. 2. Deterministic matching

The above modifications allow brokers to make matching decisions without learning the actual values, as illustrated in Figure 2. However the modified matching protocol still reveals the actual difference between the notification value and subscription values which leaks information about these values. In order to address this issue, we introduce controlled random values to the subscription and notification blinding operations so that the difference is randomized. The brokers can still make correct matching decisions by comparing which half the computed difference falls in the plaintext space, without however learning the actual difference. Figure 3 illustrates the idea behind the randomized matching.

3.2 System Architecture

We assume that logical expressions based on att/val pairs are used by subscribers to express their interest in notifications. We distinguish between two sets of attributes,

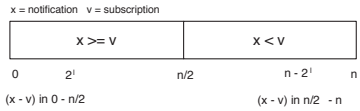


Fig. 3. Randomized matching

namely context set and static set, and an additional payload which by itself can be a set of att/val pairs. The context set contains attributes representing the subscriber context, e.g., location. The static set represents general attributes describing the message, e.g., message type. The attribute values in the context set frequently change due to changes in the context of subscribers while attribute values in the static set rarely change.

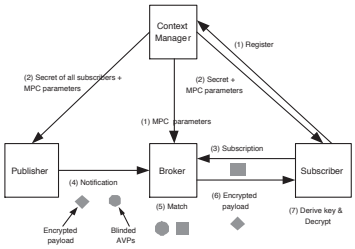


Fig. 4. System architecture

Figure 4 shows an overview of the overall system architecture and the interactions between entities. Our approach consists of four entities: publisher, subscriber, broker, and context manager.

In order to assure the privacy of subscriptions, subscribers must hide the content of their subscriptions from brokers. Further, since the attributes in the context set frequently change, subscribers must be able to update their subscriptions without contacting a TTP, referred to as the context manager. In our approach, publishers and subscribers communicate with context managers only either during the initialization phase or when security parameters change in order to obtain secrets and public parameters.

A context manager is responsible for a certain context in the system. The context is described by a set of attributes, such as location with attributes *latitude* and *longitude*. To assure the privacy of the context, subscribers must be able to issue and update subscriptions with blinded context set attribute values and publishers must be able to publish notifications and blind context set attribute values.

When publishers join the system and want to publish messages for a certain context, they first contact the responsible context manager. After successful authentication the context manager provides some security parameters and a set of secrets corresponding

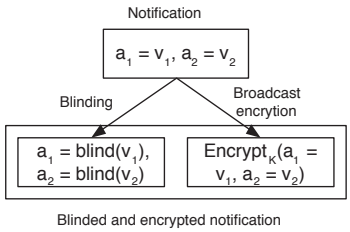


Fig. 5. Two encryptions performed by publishers

to valid subscribers to the publishers which allow them to blind and encrypt notifications. The blinding operation is performed using the modified Paillier cryptosystem described in the next section and the encryption is performed using the symmetric key generated using the AB-GKM's key generation algorithm. Figure 5 illustrates the two operations publishers perform on each notification. A similar bootstrapping process is necessary for subscribers. They contact the context manager and receive some other security parameters which allow them to issue blinded subscriptions for a certain context and secrets for identity attributes they have. Further, our scheme allows the subscribers to update their subscriptions within a context without contacting the context manager.

Brokers only receive blinded and encrypted notifications, and blinded subscriptions. When they receive a notification, they execute the matching operation on the blinded notification and the blinded subscriptions to make forwarding decisions. If a notification matches a subscription, the broker strips the blinded portion of the notification and sends only the encrypted notification to matching subscribers. It should be noted that brokers do not learn the individual notification and subscription values during matching operations.

Once subscribers receive encrypted notifications, using the secrets obtained from the context manager during the initialization phase, they can derive the decryption key using the AB-GKM's key derivation algorithm. The AB-GKM scheme makes sure that only valid subscribers can derive the key and hence decrypt notifications. Since subscribers are not given decryption keys during the registration and they must dynamically derive the keys, our approach can efficiently handle subscriber revocations and additions as well as access control policy changes without affecting the existing subscribers.

3.3 Trust Model

We consider threats and assumptions from the point of view of publishers and subscribers with respect to third-party brokers. We assume that brokers are honest but curious; they perform pub/sub operations correctly, but are curious to know the notifications and subscriptions. In other words, brokers are not trusted for the confidentiality of the notifications and subscriptions. The context manager is fully trusted. Publishers are trusted to keep the secrets obtained from the context manager confidential and to

perform notification blinding and encryption as specified. Subscribers are not trusted in our system. They can decrypt encrypted notifications only if they have valid credentials. Brokers may collude with one another as well as with malicious subscribers.

4 Modified Paillier Cryptosystem

In this section we provide the details of our modified Paillier cryptosystem.

4.1 Making μ Public

Recall that in the original Paillier cryptosystem, the tuple (λ, μ) is the private key. However, μ does not need to be private since it is hard to decrypt an encrypted message by only knowing μ. In order to decrypt, one needs to know both λ and μ. It can be shown that if a probabilistic polynomial time (PPT) adversary can obtain λ from μ, it can solve the discrete logarithm problem (DLP). Since DLP is a known hard problem, it is hard to obtain λ from μ. Hence, we can make μ public while achieving the same security guarantees as the unmodified Paillier cryptosystem. We take advantage of this fact in order to shift the computation towards encryption and make decryption light weight.

4.2 Shifting the Computation

With the modification in Section 4.1, the new public and private keys are (n, g_p, μ) and λ respectively.

First, we modify the Paillier cryptosystem so that anyone can decrypt using the new public key, but only those holding the private key can encrypt. This is similar to how digital signatures work. The following equations show the modifications to the encryption and decryption algorithms.

Encryption $E'(m, r, \lambda)$

$$E'(m, r, \lambda) = E(m, r)^{\lambda} = g_p^{m\lambda} \cdot r^{n\lambda} \pmod{n^2} = c.$$

Decryption $D(c)$

$$D(c) = L(c \pmod{n^2}) \cdot \mu \pmod{n}.$$

It should be noted that one can perform all the homomorphic operations on the modified Paillier cryptosystem similar to the unmodified Paillier cryptosystem as the above modification only shift the computation from decryption to encryption.

4.3 Computing Differences (but not Individual Values)

With the shift of computation described in Section 4.2, anyone can find the difference by simply decrypting each value. However, such an approach does not assure the privacy of individual values. Therefore, we introduce an additional parameter to the encryption

operation in order to allow one to compute the difference while at the same time not allowing the decryption of individual values.

Assume that there are two values x_1 and x_2. We perform the following modification to the encryption operation so that a decryptor can learn the difference $(x_1 - x_2)$ without learning either x_1 or x_2. We call the modified encryption as blinding operation.
Encryption $E''(x_1, x_2)$

$$x_1' = g^t \cdot E'(x_1, r_1) \pmod{n^2}$$
$$x_2' = g^{-t} \cdot E'(-x_2, r_2) \pmod{n^2}$$
Output x_1' and x_2'.

Notice that even though the decryptor knows μ, it can decrypt neither x_1' nor x_2' as they are modular multiplied with g^t and g^{-t} respectively. Due to the additive homomorphic property, the following holds: $x_1' \cdot x_2' = E'(x_1 - x_2, r_3)$.

Since the multiplication of x_1' and x_2' cancels the blinding parameters, anyone can compute the difference as follows using the public key of the modified Paillier cryptosystem: $D(x_1' \cdot x_2') = x_1 - x_2$.

4.4 Allowing Comparison

Recall that in Section 3.1 we introduced the notion of domain size which is much smaller than the plaintext space of the Paillier cryptosystem. According to our assumptions, $0 \leq x_1, x_2 \leq 2^l$ where l is the domain size and $2^l << n$. Let the difference of x_1 and x_2 be d. Due to the restriction on the domain size, d is either between 0 and 2^l or $n - 2^l$ and n. We use this fact to compare the numbers. As shown in Figure 2, if $d \leq 2^l$, then $x_1 \geq x_2$ and if $d > n - 2^l$, then $x_1 < x_2$.

During the above comparison process, the party performing the comparison learns the difference d which leaks certain information about the actual values. Hence, the comparison is not privacy preserving. We introduce a technique to randomize the difference so that it is difficult to learn the difference yet the party can learn the exact comparison result.

Notice that in the above calculation, we only utilize a small range of the plaintext space to make the comparison decision. We utilize the unused space in the plaintext space in the above calculation to randomize the difference while still allowing one to make the correct matching decision. As shown in Figure 3, the key idea is to expand the difference from $0 - 2^l$ to $0 - n/2$ and $(n - 2^l) - n$ to $n/2 - n$ by introducing controlled random values to the encryption operation. We introduce two random values r_p and r_q during the encryption operation shown below:

$$x_1'' = g^t \cdot E'(x_1, r_1)^{r_p} E'(r_q) \pmod{n^2}$$
$$x_2'' = g^{-t} \cdot E'(-x_2, r_2)^{r_p} \pmod{n^2}.$$

The decryption results in the following output:

$$D(x_1'' \cdot x_2'') = r_p(x_1 - x_2) + r_q = d'$$

r_p and r_q are randomly selected so that $d' \leq n/2$ if $x_1 \geq x_2$ and $d' > n/2$ if $x_1 < x_2$. Each time a party performs the comparison it gets a different d' due to the random values and thus the difference preserves the privacy of the individual values under comparison.

5 Privacy-Preserving Brokering Scheme

In this section, we describe in detail our approach to construct a privacy preserving context-aware publish subscribe system using the modified Paillier cryptosystem presented in Section 4 and the AB-GKM scheme.

As introduced in Section 3, there are four entities in our system: context manager, publisher, subscriber, and broker. The context manager acts as a TTP and generates the parameters for the modified Paillier cryptosystem and manages secrets obtained by the SecGen algorithm of the AB-GKM scheme to subscribers based on the identity attributes they possess. The context manager maintains a set of contexts $\mathcal{C}$ and a set of secrets issued to subscribers. Each context $C_i \in \mathcal{C}$ is a tuple of the following form: $C_i = \langle \lambda_i, \mu_i, t_i, r_i \rangle$, where λ_i and μ_i are Paillier parameters for E' and D' algorithms. t_i and r_i are random values.

Brokers match notifications with subscriptions within the same context only. μ_i values are public. λ_i and t_i values are private to the context manager.

5.1 Subscriber Registration

Each subscriber registers with the context manager. Let the context of a random subscriber be C_i. During the registration, the subscriber receives the following values from the context manager: $E'(-r_i)$, $E'(-1)$, and $g^{-t_i} \cdot E'(-r_i)$.

These parameters are used by the subscriber to blind subscriptions. Since μ_i is public, the subscriber may decrypt $E'(-r_i)$ using D' and obtain r_i. However, the subscriber can recover neither g^{-t_i} nor t_i from $g^{-t_i} \cdot E'(-r_i)$.

Using the SecGen algorithm of the AB-GKM scheme, each subscriber$_i$ also obtains secrets s_{ij} for each identity attribute j they possess from the context manager. These secrets are later used to derive the decryption key using the key derivation (KeyDer) algorithm of the AB-GKM scheme and decrypt notifications.

It should be noted that the identity attributes are not revealed to the context manager in plaintext as the SecGen algorithm internally utilizes the OCBE protocols. Thus the privacy of the identity attributes are preserved from the context manager.

5.2 Publisher Registration

Each publisher also registers with the context manager. Let the context of a random publisher be C_i. During the registration, the publisher receives the following values from the context manager: $E'(r_i)$, $E'(1)$, and $g^{t_i} \cdot E'(r_i)$.

Similar to subscriber registration, these parameters are used by the publisher to blind notifications. Since μ_i is public, the publisher may decrypt $E'(r_i)$ using D' and obtain r_i. Notice that the context manager may provide $E'(1)$ and r_i, and allow the

publisher to compute $E'(r_i)$ homomorphically instead of providing the value directly. Also, notice that the publisher can recover neither g^{t_i} nor t_i from $g^{t_i} \cdot E'(r_i)$.

In addition to the above modified Paillier cryptosystem parameters, each publisher also obtains the set of secrets issued to subscribers using the SecGen algorithm. These secrets are used to selectively encrypt notifications based on the identity attributes that subscribers possess. The publisher first uses the key generation (KeyGen) algorithm of the AB-GKM scheme to generate the encryption key based on these secrets and then encrypts the notifications using the generated key. Notice that these secrets do not reveal the actual identity attributes of subscribers to publishers. Thus the identity attributes of subscribers are preserved from publishers as well.

5.3 Notifications

Assume that a publisher wants to publish a notification for the attributes a_1 and a_2 with values v_1 and v_2 respectively. The publisher first blinds v_1 and v_2 to create v_1' and v_2' respectively using the modified Paillier cryptosystem presented in Section 4. We show the blinding operation for a general value v below.

$$\begin{aligned} v' &= g^{t_i} \cdot E'(r_i) \cdot E'(r_i(v-1)) \cdot E'(r_v) \\ &= g^{t_i} \cdot E'(r_i v + r_v), \end{aligned}$$

where r_v is a controlled random value selected by Pub.

$E'(r_i(v-1))$ is homomorphically computed using $E'(r_i)$. Notice that this value can be computed efficiently using fast multiplication.

Based on the ACP and the secrets issued to subscribers, the publisher generates the encryption key k using the KeyGen algorithm of the AB-GKM scheme. It then encrypts the payload of the notification ($a_1 = v_1, a_2 = v_2$) using the key k. We denote the encrypted payload as $E_k(payload)$. The publisher sends the blinded and encrypted notification (($a_1 = v_1', a_2 = v_2'$), $E_k(payload)$) to brokers. Notice that brokers cannot decrypt any of the blinded values as well as the encrypted payload.

An advantage of having two sets of encrypted values for each notification is that our approach allows to perform privacy preserving matching and enforce fine-grained ACPs independently.

5.4 Subscriptions

Assume that a subscriber wants to subscribe for the attribute a_1 with the value x. The subscriber blinds x and creates x' as follows:

$$\begin{aligned} x' &= g^{-t_i} \cdot E'(-r_i) \cdot E'(r_i(1-x)) \\ &= g^{-t_i} \cdot E'(-r_i x) \end{aligned}$$

$E'(r_i(1-x))$ is homomorphically computed using $E'(-r_i)$.

The tuple (a, x', α), where $\alpha = \{<, >\}$, is sent to brokers. Similar to the blinded notifications, notice that brokers cannot decrypt x'.

The tuple (a, x', α) represents a single attribute condition and we call such a subscription an *atomic subscription*. A subscription, in general, can be a Boolean expression over a set of atomic subscriptions and is called a *composite subscription*. Notice that the atomic subscription intentionally leaves the equality comparison operator. The motivation behind such a scheme is to further hide subscriptions and notifications from brokers. In our scheme, equality subscriptions are performed using range queries so that brokers cannot distinguish between equality subscriptions and range queries. In order to submit an equality subscription for attribute a with the value x, the subscriber submits the query $(a, x_1, >) \wedge (a, x_2, <)$, where x_1 and x_2 are the blinded values of $x - 1$ and $x + 1$. Since the blinded values are semantically secure, the conjunctive query does not reveal any information about the range and therefore brokers cannot distinguish an equality subscription from a general range subscription.

5.5 Broker Matching

For each context C_i, brokers receive μ_i. Assume that for the context C_i, a broker has received the blinded notification and subscription values v_1' and x' respectively for the attribute a_1. As mentioned above, we emphasize that the broker can decrypt neither v_1' nor x'. As described in Section 4.4, the broker computes the randomized difference d' as follows: $d' = D'(v' \cdot x') = r_i(v - x) + r_v$

It decides $v_1 > x$ if $d' \leq n/2$ and $d' \neq 0$, $v_1 < x$ otherwise. The above matching algorithm is described for an atomic subscription. Usually notifications contain more than one attribute and the broker has to match such notifications with either atomic or composite subscriptions. The matching for a composite subscription is performed by evaluating each atomic subscription in the subscription and evaluating the Boolean expression. After successful matching, the broker forwards only the encrypted payload $E_k(payload)$ to matching subscribers. Subscribers having valid credentials can derive the key k using the KeyDer algorithm of the AB-GKM scheme and access the payload of the notification.

6 Implementation and Evaluation

6.1 Implementation

We extended Apache ActiveMQ, a production strength and widely used message-oriented middleware, with our proposed mechanisms for privacy preserving pub/sub. ActiveMQ is a Java Message Service (JMS) broker which is the de-facto industry standard for messaging. JMS brokers support one-to-one communication via queues and pub/sub communication via topics. Topics provide basic content-based filtering mechanisms based on attribute/value pairs. JMS uses a hybrid message format where a message has a payload and in addition a set of attribute/value pairs (message properties) upon which the subscription filters (message selectors) are defined. Subscribers connect to JMS brokers and provide these message selectors which use a subset of SQL92 syntax. We extended

the subscription matching of ActiveMQ to support matching on blinded message selector attributes in subscriptions. To allow context-based subscriptions, consumers can issue new subscriptions upon context change and register it with the same message listener as the old subscription. The consumers can blind these new subscriptions without interaction with producers or a TTP.

6.2 Evaluation

Blinding operations introduce an overhead for brokers, publishers, and subscribers. To quantify this overhead we evaluated our implementation and compared it with pub/sub communication without blinding. We do not evaluate the payload encryption as it is evaluated elsewhere [24, 20].

Test Environment and Setup The test environment is shown in Figure 6. We used a distributed setup with two message generating clients connected to a host running our extended ActiveMQ broker. We started multiple publisher and subscriber threads per machine. When resources of Generator Client I were insufficient, the second client was used as support. We normalized the CPU utilization with respect to Generator Client I to provide an intuitive illustration.

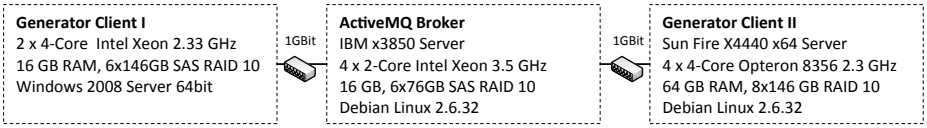
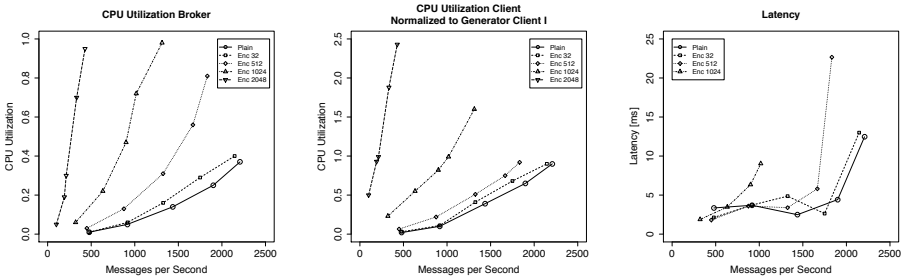


Fig. 6. Test Environment



(a) CPU Broker (b) CPU Client (c) Latency

Fig. 7. Results CONSTANT Scenario: CPU Utilization and Latency for Different Message Rates and Blinding Strength

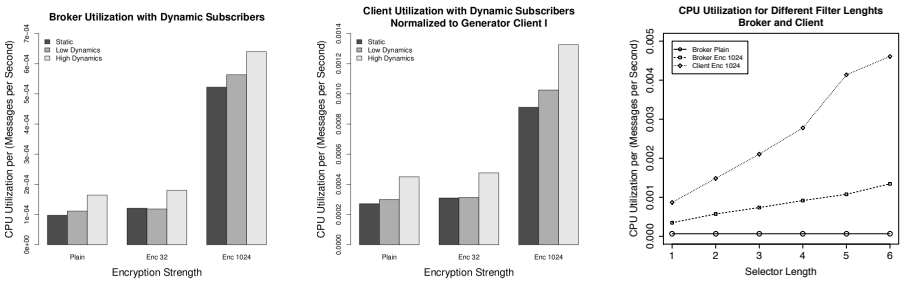
In our test case the message selectors used by subscribers required an equality check on a message property. For unencrypted matching the following selector scheme was used: `bigIntProp1 = <Random Big Integer>`. Since our blinding scheme does not allow direct equality checks the following semantically equivalent selector scheme was used for blinded messages: `bigIntProp1 > Enc(<Random Big Integer>-1) AND bigIntProp1 < Enc(<Random Big Integer>+1)`. We chose an equality check since this is a common case in pub/sub systems. It is also the worst case for blinded subscriptions. Publishers have to perform two blinding operations per message, subscribers have to perform two blinding operations per subscription, and brokers have to perform two matching calculations per message. A simple > or < comparison requires half the operations and reduces the overhead compared to unencrypted matching.

We used random numbers as message payload and to construct subscription filters. The supported domain size for encrypted properties was 100 bits. Inside the broker a matching check was performed for each message. We measured throughput, CPU utilization, and latency for unencrypted messages (plain) and for blinded messages with key lengths of 32, 512, 1024, and 2048 bit. We evaluated three scenarios:

- CONSTANT: Constant number of publishers and subscribers per test. Between tests the number of publishers and subscribers was increased resulting in an increased message rate.
- DYNAMIC: Subscribers leave and join the system at a certain rate to simulate context changes and user churn. In the *low dynamics* scenario, 2 subscribers leave and join per second. In the *high dynamics* scenario 20 subscribers leave and join per second. The joins and leaves add additional load to a *static* configuration with a constant rate of subscribers and consumers.
- COMPLEXITY: Subscribers use different message selector lengths. They subscribe with a disjunction of equality checks for up to 6 attribute values.

Results Discussion Figures 7a, 7b, and 7c show the results for the CONSTANT scenario. The CPU utilization at broker and client increases with the message rate as well as with the blinding key length. An increase in latency for blinding scenarios was below measurement variance at low loads. With increasing CPU utilization on broker and clients, the latency increases and shows the typical rapid increase when the systems reach their limit. With a key length of 1024 bit, which can be assumed to be secure, about 1200 messages per second can be handled by the broker. To produce this load, two generator machines are necessary. However, the utilization of the client is only secondary since typical real-world applications have producers and consumers spread across many machines.

Figures 8a and 8b show the CPU utilization for the DYNAMIC scenario. Issuing a new subscription requires to blind subscription values on client side. This overhead of new subscriptions is quantified in this evaluation. The CPU utilization is shown per messages per second. During the experiments we measured message rate and CPU utilization for the Static, Low -, and High Dynamics configuration. To allow a direct comparison independent of different message rates, we show the CPU utilization ratio.



(a) DYNAMIC: CPU Broker (b) DYNAMIC: CPU Client (c) COMPLEXITY: Different Selector Length

Fig. 8. Results DYNAMIC and COMPLEXITY Scenarios: Frequent Re-Subscriptions and Different Filter Lengths

The results show that the additional overhead of joining subscribers is not the factor that dominates CPU utilization at the broker. Further, the increase occurs independent of blinding which shows that the overhead of joining is inherent to the broker. On the client side CPU utilization is increased by about one third in the High Dynamics configuration compared to the Static configuration. No observable difference in the ratio of this increase can be found between the blinded and plain configurations. This shows that the CPU utilization produced by subscription operations is not dominated by blinding operations.

The results for the COMPLEXITY scenario are shown in Figure 8c. For unencrypted subscriptions an increase in CPU utilization is not observable. For blinded subscriptions the CPU utilization of the broker increases slightly with increasing complexity. The utilization on the client side increases faster since for each message all attributes have to be blinded, but the broker does not necessarily evaluate the whole message selector. Since a disjunction is used as selector a matching subexpression makes the evaluation of other subexpressions obsolete. The steeper increase in client CPU utilization from filter length 4 to 5 was caused by the activation of the second generator client. The second generator client was necessary to keep the publication rate up. The CPU utilization on this second machine includes basic operating system and Java virtual machine operations which occur as a one time effect in the results.

Overall the evaluation shows that the overhead in terms of CPU utilization of privacy preserving pub/sub is well observable. We also monitored network and memory utilization during the test runs; both were no limiting factors for performance. Thus, the throughput was limited by CPU processing power and privacy preserving pub/sub will benefit from modern multi-core systems with high processing power. It is further possible to extend our scheme and implementation to multiple brokers to build a scalable and secure pub/sub infrastructure. We think the throughput requirements of many applications can then be fulfilled and communication can be secured efficiently by our scheme.

7 Related Work

In this section, we compare our approach with existing work on secure content based pub/sub systems, privacy preserving location services and search over encrypted data.

Secure content based pub/sub systems:
Privacy and confidentiality issues in pub/sub systems have long been identified [28], but little progress has been made to address these issues in a holistic manner. Most of prior work on data confidentiality techniques in the context of content based pub/sub systems is based on the assumption that brokers are trusted with respect to the privacy of the subscriptions by subscribers [2, 27, 16]. However, when such an assumption does not hold, both publication confidentiality and subscription privacy are at risk. Further, such approaches limit brokers' ability to make routing decisions based on the content of the messages and thus fail to provide a pub/sub system as expressive as a pub/sub system that does not address security or privacy issues. Approaches have also been proposed to assure confidentiality/privacy in the presence of untrusted third-party brokers. These approaches however suffer from several limitations [23, 26, 14, 7]: inaccurate content delivery, because of the limited ability of brokers to make routing decisions based on content; weak security protocols; lack of privacy guarantees. For example, some of these approaches are prone to false positives, that is, sending irrelevant content to subscribers.

Privacy preserving location services:
In privacy preserving location services, the location of a mobile entity that requests the service is transformed to a cloaked location box that meets the given location privacy metrics. Existing techniques can be categorized into two groups: *spatial cloaking* [10, 15] and *transformation based matching* [11] techniques. Special cloaking techniques enlarge the mobile entity's location so that the location service cannot identify the exact location. However, such techniques result in a high computation and communication cost as the service has to return many irrelevant results. Transformation based matching techniques apply a one-way transformation to the query and the results so that the location service does not learn the exact location as it does not know the transformation. However, existing techniques fail to perform accurate matching.

Search over encrypted data:
Search in encrypted data is a privacy-preserving technique used in the *outsourced storage model* where a user's data are stored on a third-party server and encrypted using the user's public key. The user can use a query in the form of an encrypted token to retrieve relevant data from the server, whereas the server does not learn any more information about the query other than whether the returned data matches the search criteria. There have been efforts to support simple equality queries [25, 4] and more recently complex ones involving conjunctions and disjunctions of range queries [5]. These approaches cannot be applied directly to the pub/sub model.

8 Conclusions

We proposed an approach to construct a privacy preserving context-based pub/sub system. Our approach assures the confidentiality of notifications and subscriptions from

third-party brokers while allowing the brokers to perform matching operations. Further, publishers are able to enforce fine grained ACPs over encrypted notifications. Our solution is based on a modified Paillier cryptosystem and a recent group key management scheme.

Based on the security information obtained from context managers, subscribers blind their subscriptions and issue them to third-party brokers. Similarly, publishers blind and encrypt notifications and publish them to the brokers. Brokers are able to perform matching operation on blinded notifications and subscriptions and forward encrypted notifications to matching subscribers. A subscriber can access an encrypted notification only if it has valid attribute credentials. Unlike the existing approaches, in our approach, publishers and subscribers are able to generate notifications and subscriptions without contacting a TTP. We implemented our approach in ActiveMQ and the experimental results show that our approach is practical and efficient.

In our approach subscribers choose arbitrary brokers to subscribe in order to preserve the privacy. However, such an approach makes the message routing through the broker network inefficient. We plan to investigate the trade-off between subscriber privacy and the message routing efficiency.

References

1. S. Berkovits. How to broadcast a secret. In *EUROCRYPT 1991*, pages 535–541.
2. E. Bertino, B. Carminati, E. Ferrari, B. Thuraisingham, and A. Gupta. Selective and authentic third-party distribution of XML documents. *IEEE TKDE*, 16(10):1263–1278, Oct. 2004.
3. E. Bertino and E. Ferrari. Secure and selective dissemination of XML documents. *ACM TISS*, 5(3):290–331, 2002.
4. D. Boneh, G. Crescenzo, R. Ostrovsky, and G. Persiano. Public-key encryption with keyword search. In *EUROCRYPT 2004*.
5. D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. *CRYPTO 2007*, pages 535–554.
6. G. Chiou and W. Chen. Secure broadcasting using the secure lock. *IEEE TSE*, 15(8):929–934, Aug 1989.
7. S. Choi, G. Ghinita, and E. Bertino. A privacy-enhancing content-based publish/subscribe system using scalar product preserving transformations. In *DEXA 2010*.
8. G. Cugola, A. Margara, and M. Migliavacca. Context-aware publish-subscribe: Model, implementation, and evaluation. In *ISCC 2009*, pages 875–881.
9. P. Eugster, P.A. Felber, R. Guerraoui, and A. Kermarrec. The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2):114–131, 2003.
10. B. Gedik and L. Liu. Location privacy in mobile systems: A personalized anonymization model. In *ICDCS 2005*, pages 620–629.
11. A. Khoshgozaran and C. Shahabi. Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy. In *SSTD 2007*, pages 239–257.
12. J. Li and N. Li. OACerts: Oblivious attribute certificates. *IEEE TDSC*, 3(4):340–352, 2006.
13. G. Miklau and D. Suciu. Controlling access to published data using cryptography. In *VLDB 2003: Proceedings of the 29th international conference on Very large data bases*, pages 898–909, 2003.
14. K. Minami, A. J. Lee, M. Winslett, and N. Borisov. Secure aggregation in a publish-subscribe system. In *WPES 2008*, pages 95–104.

15. M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The new casper: query processing for location services without compromising privacy. In *VLDB 2006*, pages 763–774.
16. M. Nabeel and E. Bertino. Secure delta-publishing of XML content. In *ICDE, 2008*, pages 1361–1363.
17. M. Nabeel, N. Shang, and E. Bertino. Efficient privacy preserving content based publish subscribe systems. In *SACMAT 2012*.
18. Mohamed Nabeel and Elisa Bertino. Towards attribute based group key management. In *CCS 2011*.
19. Mohamed Nabeel, Elisa Bertino, Murat Kantarcioglu, and Bhavani M. Thuraisingham. Towards privacy preserving access control in the cloud. In *CollaborateCom 2011*, pages 172–180.
20. Mohamed Nabeel, Ning Shang, and Elisa Bertino. Privacy preserving policy based content sharing in public clouds. *IEEE TKDE*, 2012.
21. OpenID. `http://openid.net/` [Last accessed: July 18, 2012].
22. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT 1999*, pages 223–238.
23. C. Raiciu and D. S. Rosenblum. Enabling confidentiality in content-based publish/subscribe infrastructures. In *Securecomm 2006*, pages 1–11.
24. N. Shang, M. Nabeel, F. Paci, and E. Bertino. A privacy-preserving approach to policy-based content dissemination. In *ICDE 2010*.
25. D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *SP 2000*.
26. M. Srivatsa and L. Liu. Secure event dissemination in publish-subscribe networks. In *ICDCS 2007*, page 22.
27. M. Srivatsa and L. Liu. Securing publish-subscribe overlay services with eventguard. In *CCS 2005*, pages 289–298.
28. Chenxi W., A. Carzaniga, D. Evans, and A.L. Wolf. Security issues and requirements for internet-scale publish-subscribe systems. In *HICSS 2002*, pages 3940–3947.
29. X. Zou, Y. Dai, and E. Bertino. A practical and flexible key management mechanism for trusted collaborative computing. *INFOCOM 2008*, pages 538–546.