CERIAS Tech Report 2009-15

Privacy-Preserving Filtering and Covering in Content-Based Publish Subscribe Systems

by Mohamed Nabeel, Ning Shang, Elisa Bertino
Center for Education and Research
Information Assurance and Security
Purdue University, West Lafayette, IN 47907-2086

Privacy-Preserving Filtering and Covering in Content-Based Publish Subscribe Systems

Mohamed Nabeel, Ning Shang, Elisa Bertino
Purdue University,
West Lafayette, Indiana, USA
{nabeel, nshang, bertino}@cs.purdue.edu

Abstract

Content-Based Publish-Subscribe (CBPS) is an asynchronous messaging paradigm that supports a highly dynamic and many-to-many communication pattern based on the content of the messages themselves. In general, a CBPS system has three distinct parties - *Content Publishers* , *Content Brokers*, and *Subscribers* - working in a highly decoupled fashion. The ability to seamlessly scale on demand has made CBPS systems the choice of distributing *messages/documents* produced by *Content Publishers* to many *Subscribers* through *Content Brokers*. Most of the current systems assume that *Content Brokers* are trusted for the confidentiality of the data published by *Content Publishers* and the privacy of the subscriptions, which specify their interests, made by *Subscribers*. However, with the increased use of technologies, such as service oriented architectures and cloud computing, essentially outsourcing the broker functionality to third-party providers, one can no longer assume the trust relationship to hold. The problem of providing privacy/confidentiality in CBPS systems is challenging, since the solution to the problem should allow *Content Brokers* to make routing decisions based on the content without revealing the content to them. The problem may appear unsolvable since it involves conflicting goals, but in this paper, we propose a novel approach to preserve the privacy of the subscriptions made by *Subscribers* and confidentiality of the data published by *Content Publishers* using cryptographic techniques when third-party *Content Brokers* are utilized to make routing decisions based on the content. We analyze the security of our approach to show that it is indeed sound and provide experimental results to show that it is practical.

Index Terms

Privacy, Confidentiality, Security, Publish Subscribe, Filter, Cover

I. Introduction

Many systems, including online news delivery, stock quote/trade report dissemination and weather channels, have been or can be modeled as Content-Based Publish-Subscribe (CBPS) systems. Full decoupling of the involved parties, that is, *Content Publishers* (Pubs), *Content Brokers* (Brokers) and *Subscribers* (Subs), in time, space, and synchronization has been the key [15] to seamlessly scale these systems on demand. In a CBPS system, each Sub selectively subscribes to receive different messages with some Brokers. In the most common setting, when Pubs publish messages to some Brokers, these Brokers, in turn, selectively distribute these messages to other Brokers and finally to Subs based on their *subscriptions*, that is, what they subscribed to. These systems, in general, follow a *push based* dissemination approach, that is, whenever new messages arrive, Brokers selectively distribute the messages to Subs.

Because content represents the critical resource in many CBPS systems, its confidentiality is important. Consider the case of publishing stock market quotes where Subs pay Pub, that is the stock exchange, either for the types of quotes they wish to receive or per usage basis. In such a domain, whenever a new stock quote, referred to in general as a *notification*, is published, Brokers selectively send such a notification only to authorized Subs. Confidentiality is important here because Pubs want to make sure that only paying customers have access to the quotes. Throughout our paper, we assume that a message consists of a set of attribute-value pairs. We say that a CBPS system provides *publication confidentiality* if Brokers can neither identify the content of the messages published by Pubs nor infer the distribution of *attribute values* of the message. In the absence of *publication confidentiality*, Brokers may collect stock quotes, re-sell to others, and/or sell derived market data without any economic incentive to Pubs.

At the same time, the privacy of subscribers is also crucial for many reasons, like business confidentiality or personal privacy. We say that a CBPS system provides *subscription privacy* if Brokers can neither identify what subscriptions Subs made nor relate a set of subscriptions of a specific Sub. Consider again the stock quote example. Suppose for example that a Sub subscribes to some Brokers for receiving stock quotes characterized by certain attribute values (e.g. bid price < 2438, bid size > 1000, symbol = “MSFT”, etc.). In the absence of *subscription privacy*, such a subscription can reveal the business strategy of the Sub. Further, Brokers may profile *subscriptions* of each Sub and sell them to third parties.

Current trends in computing infrastructures like Service Oriented Architectures (SOAs) and cloud computing are further pushing brokering functions for content distribution to third-party providers. While such a strategy provides economies of scale, it increases the risk of breaches in publication confidentiality and subscription privacy. Breaches may result from malicious insiders or from platforms that are poorly configured and managed, and that do not have in place proper security techniques. It is thus essential that effective and efficient techniques for publication confidentiality and subscription privacy be devised to allow parties involved in the production and distribution of contents to take full advantages from those emerging computing infrastructures.

Privacy and confidentiality issues in CBPS have long been identified [33], 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 CBPS systems is based on the assumption that Brokers are trusted with respect to the privacy of the subscriptions by Subs [5], [31], [24]. However, when such an assumption does not hold, both publication confidentiality and subscription privacy are at risk; in the absence of subscription privacy, subscriptions are available in clear text to Subs. Brokers can infer the content of the notifications by comparing and matching notifications with subscriptions since CBPS systems must allow them to make such decisions to route notifications. As more subscriptions become available to Brokers, the inference is likely to be more accurate. It should also be noted that the above approaches restrict Brokers' ability to make routing decisions based on the content of the messages and thus fail to provide a CBPS system as expressive as a CBPS system that do 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 one or two major limitations [28], [22]: inaccurate content delivery, because of the limited ability of Brokers to make routing decisions based on content; lack of privacy guarantees. For example, these approaches are prone to false positives, that is, sending irrelevant content to Subs. In addition to these approaches, there has been research work on online subscription privacy [32], [6]. However, the model in such work is different from that of CBPS systems in that they follow a *pull based* dissemination approach and do not have third-party Brokers.

In this paper, we propose a novel cryptographic approach that addresses those shortcomings in CBPS systems. To the best of our knowledge, no existing cryptographic solution is able to protect both publication confidentiality and subscription privacy in CBPS systems without sending irrelevant content from Brokers to Subs. A key design goal of our privacy-preserving approach is to design a system which is as expressive as a system that does not consider privacy or security issues.

The main results presented in our paper are the design, security analysis, and performance evaluation of a CBPS system which supports all the publish/subscribe (PS) protocols implemented by current CBPS systems and exhibits the following properties:

- The published content is hidden from Brokers.
- The subscriptions made by Subs are hidden from Brokers.
- Both publication confidentiality and subscription privacy are assured without limiting the ability of Brokers to compare notifications with subscriptions and subscriptions with other

subscriptions.

The paper is organized as follows. Section II discusses related work. Section III overviews the CBPS model and the protocols supported by our system. Section IV provides some background knowledge about the main cryptographic primitives used and the trust model assumed in our approach. Section V provides a detailed description of the proposed protocols, and also illustrates our approach by an example. Section VI analyzes the security of the proposed scheme, whereas Section VII reports experimental results. Section VIII concludes the paper and outlines future work.

II. Related Work

In addition to the research work discussed in Section I, our work is related to research in proxy re-encryption systems [21], [3], [2], searchable encryption [30], [7], [8], secure multiparty computation [34], [16], [13] and private information retrieval [12], [18], [9], [23], [17], [25].

a) Proxy re-encryption system: In a proxy re-encryption system one party A delegates its decryption rights to another party B via a third party called a "proxy." More specifically, the proxy transforms a ciphertext computed under party A's public key into a different ciphertext which can be decrypted by party B with B's private key. In such a system neither the proxy nor party B alone can obtain the plaintext.

A direct application of the proxy re-encryption system does not solve the problem of CBPS: with the proxy as the Broker, it does not by default have the capability of selectively making content-based routing decisions. However, it might still be possible to use proxy re-encryption as a building block in the construction of a CBPS system for data confidentiality.

b) Searchable encryption: 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 [30], [7] and more recently complex ones involving conjunctions and disjunctions of range queries [8]. These approaches cannot be applied directly to the CBPS model: keyword-only search methods as in [30], [7] limit the application of the CBPS system; the approach proposed in [8] requires the search criteria to be known to every one, including the Broker, which thus cannot provide full privacy protection to both Pub and Subs.

c) Secure Multiparty Computation (SMC): SMC allows a set of participants to compute the value of a public function using their private values as input, but without revealing their individual private values to other participants. The problem was initially introduced by Yao [34]. Since then improvements have been proposed to the initial problem [16], [13]. SMC solutions rely on some form of zero-knowledge proof of knowledge (ZKPK) or oblivious transfer protocols which are in general interactive. Interactive protocols are not suitable for the CBPS model. Hence SMC solutions do not work for the CBPS model. Further, these solutions usually have a higher computational and/or communication cost which may not be acceptable for a CBPS system.

d) Private Information Retrieval (PIR): A PIR scheme allows a client to retrieve an item from a database server without revealing which item is retrieved. Approaches of PIR assume either the server is computationally bounded, where the problem reduces to oblivious transfer, or there are multiple non-cooperating servers each having the same copy. Having only two communication parties, PIR schemes are not directly applicable to the Pub-Sub-Broker architecture of the CBPS model. Moreover, similar to SMC solutions, PIR schemes in general require higher communication complexity which may not be acceptable for a CBPS system.

III. Overview

In this section we give an overview of our proposed scheme by showing the interactions between Pubs, Subs and Brokers using the privacy-preserving protocols we have designed. Unless otherwise stated, we describe our approach for one Pub, mainly for brevity. However, our approach can be trivially applied to a system with any number of Pubs. In practice, all the parties in a CBPS system are software programs that act on behalf of real entities like actual organizations or end users, and therefore many of the operations of the protocols we propose are performed transparently to real entities.

There are two types of messages in a CBPS system: *subscriptions* and *notifications*. The messages published by Pubs are referred to as *notifications*. Each message is characterized by a set of Attribute-Value Pairs (AVPs). A notification consists of two parts: the actual message in the encrypted form, which we call the *payload message*, and a set of *blinded AVPs* derived from the payload message. Without loss of generality, we assume that a payload message consists of a set of AVPs. In a blinded AVP, the value is blinded, but the attribute name remains in clear text. The blinding encrypts the value in a special way such that it is computationally infeasible to obtain the value from the blinded values, and that the blinded values are secure under chosen-ciphertext attacks. The blinded AVPs are placed in the header and the payload message is in the body of the notification. There is a one-to-one mapping between the AVPs in the payload message and the blinded AVPs. Since our scheme currently supports only equality of string and numerical attributes (e.g. symbol $=$ “MSFT”, bid size $= 10000$), and inequality of numerical attributes (e.g. bid price < 50), Pub blinds only those numerical and string attributes.

In an XML-like syntax, a notification has the following format:

```
<notification>
    <header>
        //blinded AVPs
    </header>
    <body>
        //encrypted payload message
    </body>
</notification>
```

Depending on the representation, each attribute name and its corresponding value may be interpreted differently. For example, the payload could be in a simple property-value format or a complex XML format. If the payload is in XML, attribute names can be the XPaths and values can be the immediate child nodes of XPaths.

A *subscription* specifies a condition on one of the attributes of the AVPs associated with the notifications. It is an expression of the form ($attr$, $bval_1$, $bval_2$, $bval_3$, op) where $attr$ is the name of the attribute, $bval_1, bval_2, bval_3$ are the blinded values of the actual content v and its additive inverse,[1] and op is a comparison operator in the set $\{<$, $>$ and $=\}$. All the other comparison operators are derived from the operators in this set.

Example 1: In the stock market quote dissemination system, a payload message, that is, a quote, looks like:

```
<quote>
    <symbol>MSFT</symbol>
```

[1] The additive inverse of a number $v \in \mathbb{Z}_m$ can be represented by the number $m - v$.

```
    <bid>
        <price>2328</price>
        <size>10000</size>
        ...
    </bid>
    <offer>
        <price>2355</price>
        <size>5000</size>
        ...
    </offer>
</quote>
```

The set of AVPs, as a collection of pairs,

$$\left\{ \begin{array}{ll} (\text{“/quote/symbol”}, \text{“MSFT”}), & (\text{“/quote/bid/price”}, 2328), \\ (\text{“/quote/bid/size”}, 10000), & (\text{“/quote/offer/price”}, 2355), \\ (\text{“/quote/offer/size”}, 5000) & \end{array} \right\}$$

from the payload message is blinded and placed in the header of the notification. The notification for the above quote will look like follows:

```
<notification>
    <header>
        <quote>
            <symbol>126452</symbol>
            <bid>
                <price>765482</price>
                <size>345219</size>
            </bid>
            <offer>
                <price>976294</price>
                <size>765291</size>
            </offer>
        </quote>
    </header>
    <body>
        //encrypted quote
    </body>
```

```
</notification>
```

We now present an overview of the protocols proposed in our CBPS system: Initialize, Register, Subscribe, Publish, Match (or Filter),[2] Cover, Revoke and Unsubscribe. Initialize protocol initializes the system parameters. Register protocol registers Subs with Pubs. Subscribe protocol subscribes Subs to Brokers. Publish protocol publishes notifications from Pubs to Brokers. Match protocol matches notifications with subscriptions at Brokers. Cover protocol finds relationships among subscriptions at Brokers. Revoke protocol allows Pubs to remove Subs from the system. Unsubscribe protocol allows Subs to remove their subscriptions from Brokers.

- Initialize:

  There is a set of system defined public parameters that all Pubs, Brokers and Subs use. In addition to these parameters, Pubs also generate some public and private parameters that are used for subsequent protocols and publish the public parameters. If there are several Pubs, each Pub generates its own public and private parameters.
- Register:

  Subs register themselves with Pub to obtain a *private key* and *access tokens*. An *access token* includes Sub's *identity* (id) and allows a Sub to subsequently authenticate itself to the Broker from which it intends to request notifications. An *identity* is a pseudonym that uniquely identifies a Sub in the system. A *private key* allows a Sub to decrypt the payload of notifications.
- Subscribe:

  After authenticating themselves using access tokens to Pubs, Subs receive the content in their subscriptions blinded by the corresponding Pubs. In this step, Sub performs as much computation as it can before sending the subscriptions to Pub so that the overhead on Pubs is minimized. Further, this overhead on Pubs is negligible as subscriptions are fairly stable and the rate of subscriptions is usually way less than that of notifications in a typical CBPS system. Once this step is done, Subs authenticate themselves to Brokers without revealing their identities and present these blinded subscriptions to Brokers. These subscriptions are blinded in such a way that Brokers do not learn the actual subscription criteria, that is,

[2] In this paper, we use the terms Match and Filter interchangeably.

Brokers cannot decrypt the blinded values. However, they can perform Match and Cover protocols based on the blinded subscriptions. Furthermore, no two subscriptions for the same value are distinguishable by Brokers. In order to prevent Brokers from linking different subscriptions from the same Sub, Subs may request for multiple access tokens such that all these access tokens have the same identity but are indistinguishable. For each subscription, Subs may present these different valid access tokens so that Subs' identities are further protected from Brokers.

- Publish:

  Using the counterparts of the secret values used to blind subscriptions, Pubs blind the notifications and publish them to the trusted list of Brokers. We assume Pubs are able to find the list of Brokers who are trusted to perform PS protocols correctly. A blinded notification has a set of blinded AVPs and an encrypted payload message. These notifications are blinded in such a way that Brokers do not learn actual values in the messages, but can perform Match and Cover protocols based on the subscriptions. Further, no two notifications for the same content are distinguishable by Brokers.

- Match:

  For each notification from Pubs, Brokers compare it with Subs' subscriptions. If there is a match, that is, the subscription satisfies the notification, Brokers forward the notification to the correct Subs. The outcome of the Match protocol allows Brokers to learn neither the notification nor the publication values. It also prevents Brokers from learning the distribution of the values.

- Cover:

  For each subscription received from Subs, Brokers check if *covering* relationship holds with existing subscriptions. A subscription S_1 covers another subscription S_2 if all notifications that match S_2 also match S_1. Finding covering relationships among subscriptions allows to reduce the size of the subscription tables maintained by each Broker, and hence improves the efficiency of matching. Like the Match protocol, the outcome of the Cover protocol does not allow the Brokers to learn the subscription values nor their distribution.

- Revoke:

  A Pub may decide to revoke a Sub for various reasons such as subscription expiration and misbehavior of Sub. Pub presents all the access tokens associated with Sub's identity to Brokers. Brokers remove all the subscriptions associated with these tokens. This may trigger `Cover` protocol to be executed one or more times in order to adjust the covering relationships affected by the removal of these subscriptions.

- `Unsubscribe`:

  Subs have the option of unsubscribing from some of the subscriptions they made so that they do not receive notifications matching with these subscriptions. A Sub authenticates itself with the same token it used to subscribe to Broker and request that subscription be removed. Similar to `Revoke`, this may trigger `Cover` protocol.

IV. Background

Our approach focuses on the following trust model and is based on the mathematical notions and the cryptographic building blocks described below.

A. Trust model

In the system design, we consider threats and assumptions from the point of view of Pubs and Subs with respect to third-party Brokers.

We assume that Brokers are honest but curious; they perform `Match`, `Cover`, `Subscribe`, `Revoke` and `Unsubscribe` protocols correctly, but curious to know what Pubs publish and Subs consume. In other words, they are trusted for these PS protocols but not for the content in the notifications and subscriptions nor for the privacy of Subs if they make multiple subscription requests.

Pubs are trusted to maintain the privacy of Subs. However, our approach can be easily modified to relax this trust assumption. Pubs are also trusted to correctly perform PS protocols and not to collude with any other parties.

B. Pedersen commitment

A cryptographic “commitment” is a piece of information that allows one to commit to a value while keeping it hidden, and preserving the ability to reveal the value at a later time. The *Pedersen commitment* [27] is an unconditionally hiding and computationally binding commitment scheme which is based on the intractability of the discrete logarithm problem. Other well known cryptographic schemes, like the Zero-knowledge proof of knowledge (ZKPK), are built on top of the Pedersen commitment.

Pedersen Commitment

Setup A trusted third party T chooses a multiplicatively written finite cyclic group G of large prime order $\mathfrak{p}$ so that the computational Diffie-Hellman problem is hard in G.[3] T chooses two generators g and h of G such that it is hard to find the discrete logarithm of h with respect to g, i.e., an integer x such that $h = g^x$. It is not required that T know the secret number x. T publishes $(G, \mathfrak{p}, g, h)$ as the system parameters.

Commit The domain of committed values is the finite field $\mathbb{F}_\mathfrak{p}$ of $\mathfrak{p}$ elements, which can be represented as the set of integers $\mathbb{F}_\mathfrak{p} = \{0, 1, \ldots, \mathfrak{p} - 1\}$. For a party U to commit a value $\alpha \in \mathbb{F}_\mathfrak{p}$, U chooses $\beta \in \mathbb{F}_\mathfrak{p}$ at random, and computes the commitment $c = g^\alpha h^\beta \in G$.

Open U shows the values α and β to open a commitment c. The verifier checks whether $c = g^\alpha h^\beta$.

C. Zero-knowledge proof of knowledge (Schnorr's scheme)

The *zero-knowledge proof of knowledge (ZKPK)* protocol used in this paper can be viewed a natural extension of Schnorr's scheme [29]. In our proposed approach, we use ZKPK as a privacy-preserving means of subscriber authentication to the brokers.

As in the case of the Pedersen commitment scheme, a trusted party T generates public parameters $G, \mathfrak{p}, g, h$. A Prover which holds private knowledge of values α and β can convince a $\mathsf{Verifier}$ that Prover can open the Pedersen commitment $c = g^\alpha h^\beta$ as follows.

1) Prover randomly chooses $y, s \in \mathbb{F}_\mathfrak{p}^*$, and sends $\mathsf{Verifier}$ the element $d = g^y h^s \in G$.
2) $\mathsf{Verifier}$ picks a random value $e \in \mathbb{F}_\mathfrak{p}^*$, and sends e as a challenge to Prover.
3) Prover sends $u = y + e\alpha, v = s + e\beta$, both in $\mathbb{F}_\mathfrak{p}$, to $\mathsf{Verifier}$.
4) $\mathsf{Verifier}$ accepts the proof if and only if $g^u h^v = d \cdot c^e$ in G.

D. Euler's totient function $\phi(\cdot)$ and Euler's theorem

Let $\mathbb{Z}$ be the set of integers. Let $\mathbb{Z}^+$ denote all positive integers. Let $m \in \mathbb{Z}^+$. The *Euler's totient function* $\phi(m)$ is defined as the number of integers in $\mathbb{Z}^+$ less than or equal to m and

[3] For a multiplicatively written cyclic group G of order q, with a generator $g \in G$, the *Computational Diffie-Hellman problem (CDH)* is the following problem: Given g^a and g^b for randomly-chosen secret $a, b \in \{0, \ldots, q-1\}$, compute g^{ab}.

relatively prime to m.

Theorem 1 (Euler's Theorem): Let $m \in \mathbb{Z}^+$. If $\gcd(a, m) = 1$, then $a^{\phi(m)} \equiv 1 \pmod{m}$.

E. Paillier homomorphic cryptosystem

The *Paillier homomorphic cryptosystem* is a public key cryptosystem by Paillier [26] based on the "Composite Residuosity assumption (CRA)." The Paillier cryptosystem is homomorphic in that, by using 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. It works as follows.

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

$$\gcd\left(L(g_p^\lambda \pmod{n^2}), n)\right) = 1,$$

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

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

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

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

More specifically, the homomorphic properties of Paillier cryptosystem are:

$$D(E(m_1,r_1)E(m_2,r_2) \pmod{n^2}) = m_1 + m_2 \pmod{n},$$
$$D(g^{m_2}E(m_1,r_1) \pmod{n^2}) = m_1 + m_2 \pmod{n},$$
$$D(E(m_1,r_1)^k \pmod{n^2}) = km_1 \pmod{n}.$$

Also note that the Paillier cryptosystem described above is semantically secure against chosen-plaintext attacks (IND-CPA).

In the construction of our CBPS system, the Paillier homomorphic cryptosystem is used in a way that public and private keys are judiciously distributed among Pubs, Subs, and Brokers to that privacy is assured based on homomorphic encryption. A detailed description of the construction will be presented in Section V.

V. Proposed Scheme

In this section, we provide a detailed description of the privacy preserving CBPS system we propose. As introduced in Section III, the system consists of 8 protocols: 1) `Initialize`, 2) `Register`, 3) `Subscribe`, 4) `Publish`, 5) `Match`, 6) `Cover`, 7) `Revoke`, and 8) `Unsubscribe`. We discuss only the first 6 protocols in this section, as the latter two are more related to subscription management which is not the focus of this paper, and has been researched extensively [10], [11], [4], [19], [20].

A. Initialize

A trusted party, which could be one of the Pubs, runs a Pedersen commitment setup algorithm to generate system wide parameters $(G, \mathfrak{p}, g, h)$. These parameters have the same meaning and purpose as mentioned in Section IV. The same party also runs a Paillier key generation algorithm to generate the parameters $(n, p, q, g_p, \lambda, \mu)$. Only Pubs know the parameters (p, q, λ) and (n, g_p, μ) are public parameters. The system parameter l is the upper bound on the number of bits required to represent any data values published, and we refer to it as *domain size*. For example, if an attribute can take values from 0 up to 500 ($< 2^9$), l should be at least 9 bits long. For reasons that will soon become clear in this section we choose l such that $2^{2l} \ll n$.[4]

[4] We use notation $a \ll b$ to denote that "a is sufficiently smaller than b."

In addition to these parameters, each Pub has a key pair (K_{pub}, K_{pri}) where K_{pri} is the private key used to sign access tokens of Subs and K_{pub} is the public key used by Brokers to verify authenticity and integrity of them. Each Pub also has a symmetric key K which it shares only with Subs and is used to encrypt the payload messages. Each Pub computes two pairs of secret values (e_m, d_m) and (e_c, d_c) such that $e_m + d_m \equiv 0 \pmod{\phi(n^2)}$, and $e_c + d_c \equiv 0 \pmod{\phi(n^2)}$, where $\phi(\cdot)$ is Euler's totient function and $e_m \neq e_c$. Note that we have $g^{e_m} g^{d_m} \equiv g^{e_c} g^{d_c} \equiv 1 \pmod{n^2}$ by Theorem 1. Pub uses e_m to blind Paillier encrypted notifications and d_m, d_c, d_c to blind Paillier encrypted subscriptions.[5] The list $\mathcal{B}$ of Brokers from which Subs may request notifications from Pub is also public. Let s be the largest number $\in \mathbb{Z}$ such that $2^s < n$. Finally, each Pub chooses two secret random values $r_m, r_c \in \mathbb{Z}$ such that $r < 2^{s-1-l}$ and $r_m \neq r_c$. This value is used to prevent Brokers from learning the distribution of the difference of the values that are being matched. In summary, $(G, \mathfrak{p}, g, h; n, g_p, \mu, K_{pub}, \mathcal{B})$ are the public parameters that all the parties know, $(p, q, \lambda, K_{pri}, r_m, r_c, (e_m, d_m), (e_c, d_c))$ are private parameters of Pubs. Note that in a practical implementation, most of these parameters can be auto-generated by a computer program which usually only requires Pub to pre-determine l depending on the domain of the content of notifications.

B. Register

Each Sub registers itself with the Pub by presenting an id (identity), a pseudonym uniquely identifying Sub. In a real-world system, registration may involve Sub presenting other credentials and/or making payment. Upon successful registration, Pub sends K, the symmetric key, to Sub.[6] During this protocol, Sub also creates its initial access token, a Pedersen commitment signed by Pub.

An access token allows a Sub to authenticate itself to the Broker from which it intends to request notifications as well as to create additional access tokens in consultation with Pub. To create the first access token, Sub encodes its id as an element $\langle \mathsf{id} \rangle \in \mathbb{F}_p$, chooses a random $a \in \mathbb{F}_p$, and sends the commitment $com(\langle \mathsf{id} \rangle) = g^{\langle \mathsf{id} \rangle} h^a$ and the values ($\langle \mathsf{id} \rangle$, a). Pub signs

[5] The "blind" operation will be introduced in Section V-C.

[6] We use a symmetric encryption algorithm in the presentation. In practice, Pubs and Subs can choose any encryption scheme, symmetric or not, to hide the payload messages in transmission. Proxy re-encryption, as mentioned in Section II, can be one of such choices.

$com(\langle \mathsf{id} \rangle)$ and sends the digital signature $K_{pri}(com(\langle \mathsf{id} \rangle))$ back to Sub. Figure 1 shows a high-level interaction between Pub and Sub for this protocol.

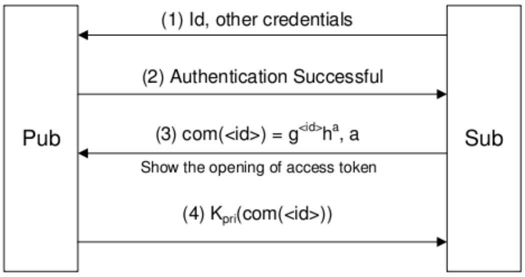
Pub
Sub
(1) Id, other credentials
(2) Authentication Successful
(3) com(<id>) = g<id>ha, a
Show the opening of access token
(4) Kpri(com(<id>))

Fig. 1. Sub registering with Pub

C. Subscribe

During this protocol, Subs inform their interests to Brokers as subscriptions. The blinded values in the subscription ($attr$, $bval(v, d_c, r_c)$, $bval(-v, e_c, r_c)$, $bval(-v, d_m, r_m)$, op) are computed with the help of Pub where the value v is the original value. Sub computes the Paillier encryption of v, $E(v)$, and that of the additive inverse $-v$, $E(-v)$ as

$$E(v) = g_p^v \cdot r_1^n \pmod{n^2},$$
$$E(-v) = g_p^{n-v} \cdot r_2^n \pmod{n^2},$$

where r_1 and r_2 are random values in $\{1, 2, \ldots, n-1\}$. The first two blinded values in the subscription are used by Broker for `Cover` protocol and the third on for `Match` protocol.

Sub sends $E(v)$ and $E(-v)$ to Pub who computes $bval(v, d_c, r_c)$, the blinded value of v, $bval(-v, e_c, r_c)$ and $bval(-v, d_m, r_m)$, the blinded values of $-v$, using d_c, e_c, d_m, respectively, where the "blinding" operation is

$$bval(x, y, r) = g^y \cdot (E(x))^{r\lambda} \pmod{n^2}, \tag{3}$$

and $d_c, e_c, d_m, r_c, r_m, \lambda$ are private parameters of Pub generated during `Initialize`. Sub then sends the computed $bval(v, d_c, r_c)$, $bval(-v, e_c, r_c)$ and $bval(-v, d_m, r_m)$ back to Sub. Figure 2 shows a high-level interaction between Pub, Sub and Broker for this protocol.

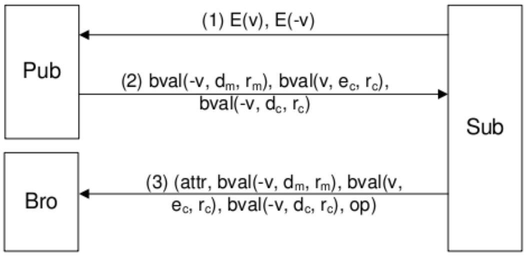
Pub
Bro
Sub
(1) E(v), E(-v)
(2) bval(-v, d_m, r_m), bval(v, e_c, r_c), bval(-v, d_c, r_c)
(3) (attr, bval(-v, d_m, r_m), bval(v, e_c, r_c), bval(-v, d_c, r_c), op)

Fig. 2. Sub authenticating itself to Broker

Before subscribing to messages, as Figure 3 illustrates, Subs must authenticate themselves to Brokers. Sub gives a zero-knowledge proof of knowledge (ZKPK) of the ability to open of the commitment $com(\langle \mathsf{id} \rangle)$ signed by Pub:

$$\text{ZKPK}\{(\langle \mathsf{id} \rangle, a) : com(\langle \mathsf{id} \rangle) = g^{\langle \mathsf{id} \rangle} h^a\}$$

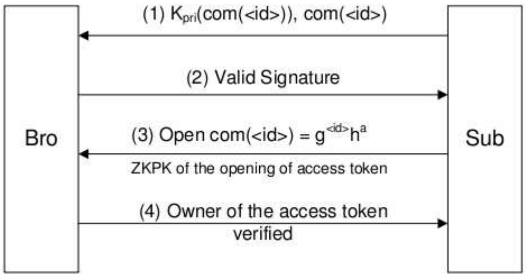
Bro
Sub
(1) K_{pri}(com(<id>)), com(<id>)
(2) Valid Signature
(3) Open com(<id>) = g^{<id>}h^a
ZKPK of the opening of access token
(4) Owner of the access token verified

Fig. 3. Sub authenticating itself to Broker

If the ZKPK is successful, Sub may submit one or more subscriptions. Notice that the ZKPK of the commitment opening does not reveal the identity of the Sub. Further, Sub may use different access tokens by having different random a values for different subscriptions to prevent Brokers from linking its subscriptions to one access token.

Example 2: A Sub wants to get all the notifications with bid price greater than 22. The subscription has the format (“/quote/bid/price”, 346213, 152311, 453280, $>$) where the second and third parameters are the blind values of 22 and -22, respectively, for `Cover` protocol to use, and the fourth is the blinded value of -22 for `Match` protocol to use.

D. Publish

Using e_m, the counterpart of d_m which is used to blind subscriptions for `Match` protocol, and other private parameters, Pubs blind the notifications using formula (3), and publish them to $\mathcal{B}$. A notification has a set of blinded AVPs and an encrypted payload message. These notifications are blinded in such a way that Brokers do not learn actual content in the notifications, but they can perform `Match` and `Cover` protocols based on the notifications.

The header of a notification is a list of blinded AVPs. Similar to the payload message, these blinded AVPs can be represented in different formats such as Java properties or XML representations.

For an illustration purpose, let us assume these AVPs are numbered from 1 to t, where t is the number of attributes of the payload message M being considered. The blinded content is formatted as follows:

$$\begin{aligned}&(attr_1, bval(x_1, e_m, r_m)),\\&(attr_2, bval(x_2, e_m, r_m)),\\&\cdots\cdots,\\&(attr_t, bval(x_t, e_m, r_m)),\end{aligned}$$

where $attr_i$ is the i^{th} attribute name, $bval(x_i, e_m, r_m)$ is the corresponding blinded value with the original value bing x_i and e_m, r_m are secret parameters of Pub.

Pub computes each $bval(x_i, e_m, r_m)$ ($i = 1, 2, \ldots, t$) as follows and publishes to $\mathcal{B}$ as a single notification along with the encrypted payload message M, $K(M)$.

$$bval(x_i, e_m, r_m) = g^{e_m} \cdot (E(x_i))^{r_m \lambda} \pmod{n^2}$$

E. Match

For each notification from Pubs, Brokers compare it with Subs' subscriptions to make routing decisions. We explain the `Match` operation for one attribute in the message, but it can be naturally extended to perform on multiple attributes. If at least one of the attributes in the

TABLE I

MATCHING DECISION

diff	Decision
0	$x = v$
$< 2^{s-1}$	$x > v$
$> 2^{s-1}$	$x < v$

message matches, we say that the subscription matches the notification, and in this case Brokers forward the notification to the corresponding Subs.

Let the blinded values be $bval(x, e_m, r_m)$ and $bval(-v, d_m, r_m)$ that Broker has received from Pub and Sub, respectively, for an attribute $attr$ with subscription value being v and notification value being x. Broker computes the following value $diff$ and then makes the matching decision based on Table I:

$$diff = L(bval(x, e_m, r_m) \cdot bval(-v, d_m, r_m) \pmod{n^2}) \cdot \mu \pmod{n},$$

where L, μ are Paillier parameters. The above computation gives the value of $r_m \cdot (x - v)$. When the system initializes, the range of values is set to 2^l. The difference of any two values less than 2^l is either between 0 and 2^l if the difference is positive, or between $(n - 2^l)$ and n if the difference is negative. Notice that the values between 2^l and $(n - 2^l)$ are not used. In order to hide the difference, we take advantage of this unused range and multiply the actual difference with a random secret value r_m selected by Pub. The idea behind r_m is to expand $0 \sim 2^l$ range close to $0 \sim (n/2)$ and $(n - 2^l) \sim n$ close to $n/2 \sim n$. This still allows Broker to make correct matching decisions without resulting in false positives or negatives. The idea is illustrated in Figure 4. In order to ease the presentation, in our discussion we deal with 2^s, instead of n, where s is defined in the description of `Initialize`.

During the `Match` protocol, Broker does not learn the content under comparison. This is achieved due to the fact that without knowing λ, Broker cannot perform Paillier decryption freely, but is force to engage into the protocol described below. Not knowing the value r_m, Broker does not learn the exact difference of the two values under comparison as well.

In the following we shall show how our approach intelligently distribute Paillier parameters

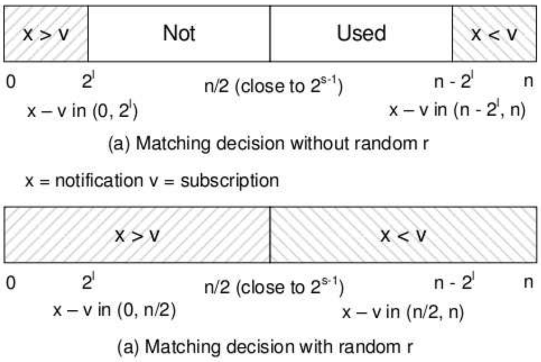

Fig. 4. Using the unused range to hide the difference

to allow Brokers to recover the blinded difference without knowing λ. Let

$$y = bval(x, e_m, r_m) \cdot bval(-v, d_m, r_m) \pmod{n^2}.$$

It can be easily checked that

$$\begin{aligned} y &= g^{e_m} \cdot (E(x))^{r_m\lambda} \cdot g^{d_m} \cdot (E(-v))^{r_m\lambda} \pmod{n^2} \\ &= g^{e_m+d_m} \cdot \{E(x) \cdot E(-v)\}^{r_m\lambda} \pmod{n^2} \\ &= (E(x-v))^{r_m\lambda} \pmod{n^2} \\ &= (E(r_m(x-v)))^{\lambda} \pmod{n^2}. \end{aligned}$$

Then

$$\mathit{diff} = L(y) \cdot \mu \pmod{n} = r_m(x-v). \tag{4}$$

F. Cover

Subscriptions are categorized into groups based on the covering relationships so that Brokers can perform `Match` protocol efficiently. For each subscription received from Subs, Brokers check if covering relationship holds within the existing subscriptions. If it exists, Broker adds the new subscription to the group with the covering subscription, otherwise a new group is created for the new subscription.

Notice that we have not used the blinded values $bval(-v, d_c, r_c)$ and $bval(v, e_c, r_c)$ in subscriptions yet. These two values are used in the Cover protocol. In what follows, we explain how the Cover protocol works.

Let S_1 and S_2 be two subscriptions for the same $attr$ and compatible op. Two op's are compatible if either both of them are of the same type or at least one of them is $=$ operation. $bval(v_1, e_c, r_c)$ and $bval(-v_1, d_c, r_c)$ refer to the so far unused blinded values of v_1 and of its additive inverse, respectively, of the subscription S_1. The blinded values $bval(v_2, e_c, r_c)$ and $bval(-v_2, d_c, r_c)$ have similar interpretations.

Broker computes one of the following two values in order to decide the covering relationship.

$$\begin{aligned} \mathit{diff}_1 &= L(bval(v_1, e_c, r_c) \cdot bval(-v_2, d_c, r_c) \\ &\quad \pmod{n^2}) \cdot \mu \pmod{n} \\ \mathit{diff}_2 &= L(bval(v_2, e_c, r_c) \cdot bval(-v_1, d_c, r_c) \\ &\quad \pmod{n^2}) \cdot \mu \pmod{n} \end{aligned} \tag{5}$$

diff_1 and diff_2 give results $r_c \cdot (v_1 - v_2)$ and $r_c \cdot (v_2 - v_1)$. The Broker uses the same table Table I that is used for making matching decision to make the covering decision. Similar to Match, Brokers does not learn the actual subscription values. Notice that due to the secret factor r_c, Brokers will not learn the actual difference of two different subscriptions made for the same attribute.

G. A Simple Example

We now walk through a simple example to demonstrate the scheme proposed earlier in this section. For simplicity and without loss of generality, we use small numbers in the presentation. In a real-world system much larger numbers will be used to match practical security requirements.

Assume that the system has 1 Pub, 1 Broker, 3 Subs and the messages have two numerical attributes $attr_1$ and $attr_2$.

Setup:

We set n = 13 bits and l = 5 bits, that is, $attr_1$ and $attr_2$ can take values $0, 1, \cdots, 31$.

We choose the following parameters for constructing the Paillier cryptosystem in this example.

$$p_p = 5, \qquad q_p = 41, \qquad n_p = 2173, \qquad n^2 = 4721929$$

$$g_p = 2, \qquad \lambda = 520, \qquad \mu = 83.$$

Pub generates e_m, d_m, e_c, d_c such that $e_m + d_m \equiv 0 \pmod{\phi(n^2)}$, and $e_c + d_c \equiv 0 \pmod{\phi(n^2)}$, where $\phi(n^2) = 4519840$:

$$e_m = 3374905, \qquad d_m = 1144935,$$

$$e_c = 502817, \qquad d_c = 4017023.$$

We have s = 12 ($2^{12} < n_p$). Pub chooses random r_m and r_c such that $r_m, r_c < 2^{s-l-1} = 64$:

$$r_m = 36, \quad r_c = 48.$$

The triples (e_m, d_m, r_m) and (e_c, d_c, r_c) are used in `Match` and `Cover` protocols, respectively.

`Subscribe`:

Assume that Sub$_1$ and Sub$_2$ make subscriptions ($attr_1 < 20$) and ($attr_1 < 18$), respectively, and Sub$_3$ makes a subscription ($attr_2 > 15$). In rest of this section, we shall show how to execute privacy preserving `Match` and `Cover` protocols using these three subscription instances.

Table II shows the two Paillier encrypted values each Sub sends to Pub.

TABLE II

ENCRYPTED VALUES FROM SUBS TO PUB

Sub	Actual v	$E(v)$	$E(-v)$
Sub$_1$	20	2209050	2600328
Sub$_2$	18	3332492	3317148
Sub$_3$	15	2515030	3069803

For each of the requests in Table II, Pub generates 3 blinded values as shown in Table III and sends to Subs.

Subs subscribe with Broker by providing the information in Table IV along with the blinded values in Table III.

TABLE III

BLINDED VALUES FROM PUB TO SUBS

Sub	$g^{d_m}E^{r_m \cdot \lambda}(-v)$	$g^{e_c}E^{r_c \cdot \lambda}(v)$	$g^{d_c}E^{r_c \cdot \lambda}(-v)$
Sub$_1$	3286610	1722651	3310307
Sub$_2$	3358319	2676598	3286404
Sub$_3$	1104918	1746554	889585

TABLE IV

SUBSCRIPTION CRITERIA OF SUBS

Sub	attr	op
Sub$_1$	$attr_1$	$<$
Sub$_2$	$attr_1$	$<$
Sub$_3$	$attr_2$	$>$

Cover:

For the three subscriptions Broker has, it can only find covering relationships for the same attribute and compatible operators. Therefore, Broker can only compare Sub$_1$'s and Sub$_2$'s subscriptions for covering relationship. Broker checks if the Sub$_1$'s subscription is greater than Sub$_2$'s to determine if the former covers the latter. From Table III, Broker multiplies Sub$_1$'s $g^{e_c}E^{r_c \cdot \lambda}(v)$ with Sub$_2$'s $g^{d_c}E^{r_c \cdot \lambda}(-v)$ and unblinds to obtains the value 96. Since $96 < 2^{s-1}$ = 1024, Broker decides that the Sub$_1$'s subscription covers the Sub$_2$'s subscription. In other words, if a notification matches the Sub$_2$'s subscription, Broker can infer that it matches the Sub$_1$'s subscription as well, without executing another Match protocol. Notice that Broker carries out a Match protocol without knowing either subscriptions in clear text.

Publish:

Pub publishes a notification to Broker. Broker has access only to the blinded values in the third column of Table V. Since Broker knows that Sub$_1$'s subscription covers Sub$_2$'s subscription, Broker first performs the Match protocol for Sub$_2$'s subscription. Broker then multiplies Sub$_2$'s blinded value $g^{d_m}E^{r_m \cdot \lambda}(-v)$ (as in Table III) with $attr_1$'s blinded value $g^{e_m}E^{r_m \cdot \lambda}(x)$ (as in Table V), and unblinds, using formula (4), to find that this value is greater than 2^{s-1} = 1024. Since Sub$_2$'s subscription matches the notification, it also matches Sub$_1$'s subscription. Thus

TABLE V

CONTENT OF THE NOTIFICATION

attr	x	$g^{e_m} E^{r_m \cdot \lambda}(x)$
$attr_1$	16	1502764
$attr_2$	10	1667912

Broker forwards the notification to both Sub_1 and Sub_2. With a similar computation, Broker finds that Sub_3's subscription does not match the notification by using Sub_3's $g^{d_m} E^{r_m \cdot \lambda}(-v)$ and $attr_2$'s $g^{e_m} E^{r_m \cdot \lambda}(x)$, and thus does not forward the notification to Sub_3. As illustrated, Broker can perform PS protocols without learning the actual content of notifications and subscriptions.

VI. Security Analysis

In this section, we analyze the security of the proposed CBPS system.

The proposed system is built upon provably secure cryptographic primitives: digital signatures, Pedersen commitment, Schnorr's zero-knowledge proof protocol, and Paillier homomorphic encryption.

A. Privacy-preserving subscription

The subscription protocol is privacy preserving in that it supports anonymous credential authentication of the Subs to Brokers. When a Sub subscribes to a Broker, it shows an access token containing a Pedersen commitment of Sub's identity attribute value $\langle \mathsf{id} \rangle$ together with a digital signature from a Pub. Broker verifies the digital signature using Pub's public key K_{pub} to make sure that the Pedersen commitment is a valid one approved by Pub. Due to the unconditional hiding property of the Pedersen commitment scheme, Broker learns nothing about the value $\langle \mathsf{id} \rangle$ from $com(\langle \mathsf{id} \rangle) = g^{\langle \mathsf{id} \rangle} h^a$. By performing a zero-knowledge proof of knowledge protocol, Sub can convince Broker that Sub knows the values $\langle \mathsf{id} \rangle$ and a, thus has the ability to open the commitment, but prevents Broker from learning the actual values. Without knowing the values $\langle \mathsf{id} \rangle$ and a, anyone without valid ownership to the access token cannot open the commitment. This provides a mechanism to defend identity theft. In such a way, the combined use of digital signatures and the ZKPK technique realizes a privacy-preserving authentication.

B. Privacy-preserving matching and covering

Match and Cover protocols are privacy preserving in that while Brokers are performing matching and covering operations correctly, they do not learn the actual values in Subs' subscriptions or Pubs' notifications.

To see that Match preserves Pub and Sub's privacy, we look at the way Paillier homomorphic encryption is used. When Sub subscribes, Broker gets a subscription specified with blinded values $bval(v, d_c, r_c)$, $bval(-v, e_c, r_c)$, and $bval(-v, d_m, r_m)$ from which the actual value v cannot be recovered with only the public parameters. Note that Broker even may not be able to feed these blinded values into formula (1) in an attempt to recover the unblinded values, because in general the blinded values are not in the domain S_n of function $L(\cdot)$ (see Section IV-E). In this way Broker is forced to follow the Match protocol as specified and make matching decisions using Table I.

Similarly, in Cover protocol, although Broker is able to perform operation as in formula (5) to obtain $r_c \cdot (v_1 - v_2)$ or $r_c \cdot (v_2 - v_1)$, then use Table I to make covering decisions, it cannot perform decryption to get either v_1 or v_2 from the blinded values. In this way, Subs' subscription privacy is protected.

Note that having the same r value over a long period of time may allow Broker to gather enough information to discover r, thus the real difference of two unblinded values, by computing the greatest common divisor of the values returned from Match or Cover protocol. Therefore, we suggest that Pubs change their r values periodically and notify involved Brokers with the change. Determination of the frequency of the update of r values depends on various issues like the message exchange rate and the number of subscriptions, and a detailed discussion is out of the scope of this paper. We want to remark that periodically updating r at Pubs makes it harder for Brokers to discover the real differences, but it does not completely eliminate the attack via computation of the gcd, which our scheme currently does not address. In any case, the actual values in Pub notifications and Sub subscriptions are kept secret from Brokers.

VII. Experimental Results

In this section, we present experimental results for various operations of the protocols in our system. We have built a prototype system in Java that incorporates our techniques for privacy preserving Match and Cover protocols as described in Section V.

The experiments were performed on an Intel® Core^{TM} 2 Duo CPU T9300 2.50GHz machine running GNU/Linux kernel version 2.6.27 with 4 Gbytes memory. We utilized only one processor for computation. The code is built with Java version 1.6.0. along with Bouncy Castle lightweight APIs [1] for most cryptographic operations including the symmetric-key encryption. The Paillier cryptosystem is implemented as in the paper [26], except that we modified the algorithms to fit our scheme.

In our experiments we vary values of n in Paillier cryptosystem and the domain size l, and fix the parameters for Pedersen commitment generation, digital signature generation/verification, zero-knowledge proof of knowledge protocol, and symmetric key encryption/decryption which have already been evaluated elsewhere. However, we compare our protocol results with these well established computations to show that our approach is efficient and practical. In all our experiments we only measure computational cost, and assume the communication cost to be negligible. Note that in a distributed setting the communication cost can be an important factor. However its evaluation is beyond the scope of our work. All data obtained by our experiments correspond to the average time taken over 1000 executions of the protocols with varying values for the bit length of n in the Paillier cryptosystem and the domain size l.

TABLE VI

AVERAGE COMPUTATION TIME FOR GENERAL OPERATIONS

Computation	Time (in ms)
Create access token (Sub)	4.21
Open access token (Pub)	4.17
Sign access token (Pub)	4.10
Verify token signature (Broker)	0.36
ZKP of access token (Sub)	4.18
ZKP of access token (Broker)	6.31
Encrypt payload message (Pub)	34.56
Decrypt payload message (Sub)	0.36

Table VI shows the average running time for various operations for which we kept the parameters constant. Access token creation, opening, signing are performed during `Register` protocol and based on Pedersen commitment scheme. Pub signs the access token using SHA-1 and RSA with 1024-bit long private key K_{pri}. Verification of the signature on the access token using the public key K_{pub}, and the ownership proof of the access token via the ZKPK are performed during `Subscribe` protocol. Zero-Knowledge Proof (ZKP) protocols are generally considered time consuming, but in our approach ZKP computation is comparable to other operations in the system, in that it takes merely a few milliseconds. For the experiments, we set the payload size to 4 Kbytes and used AES-128 as the symmetric key algorithm. These performance results demonstrate that the constructs we use and the computations are very efficient.

In the experiment shown in Figure 5, we vary the bit length of n in the Paillier cryptosystem. Figure 5 shows the time to generate blinded subscriptions and notifications whose values are less than 2^l where l, the domain size, is fixed at 100, a reasonably large value. The time to generate blinded values increases as the bit length of n increases, but even for large bit lengths, it takes only a few milliseconds. The time required to blind subscription is split into two tasks with Sub performing the encryption and Pub performing the blinding, but to blind notifications, Pub performs both operations as one task.

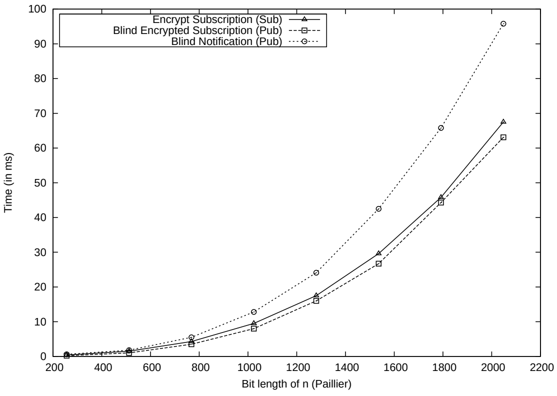

Fig. 5. Time to blind subscriptions and notifications for different bit lengths of n

We measure in our experiment the performance impact on blinding when l, the domain size, is changed. We fix n to be of length 1024 bits and measure the time to blind subscriptions and notifications for $l = 10, 20, \cdots, 100$. As shown in Figure 6, the domain size does not significantly affect the performance of the blinding operations. Further, as indicated by both Figures 5 and 6, the time for either component of the subscription blinding is less than that for notification blinding. Since for each subscription, the overhead at Pub is less compared to the time required to blind a notification, our decision to blind part of the subscription at Pub is comparable to blinding additional notifications.

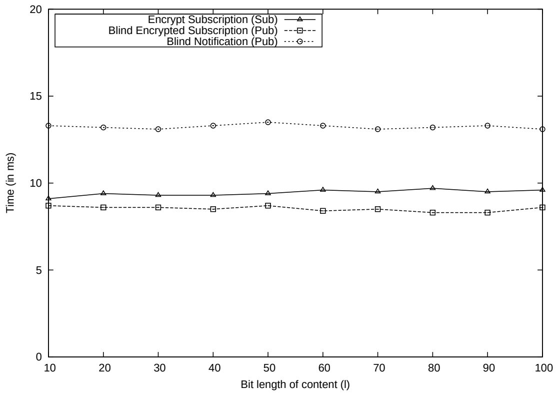

Fig. 6. Time to blind subscriptions and notifications for different l

In a CBPS, Match is the most executed protocol. Hence, it should be very efficient so as not to overload Brokers. For each Subscribe, Revoke and Unsubscribe, Broker may need to invoke the Cover protocol and, therefore, we want to have a very efficient Cover protocol as well. In the following two experiments, we observe the time to perform these protocols.

Figure 7 shows the execution time of Match and Cover protocols as the bit length of n in the Paillier cryptosystem is changed while the domain size l is fixed at 100 bits. The time for both protocols increases approximately linearly with the bit length of n. Note that they take only a fraction of a millisecond (less than 100 microseconds) even for large bit lengths of n. This indicates that our Match and Cover protocols are very efficient for large bit lengths of n.

Figure 8 shows the time to execute Match and Cover protocols as the domain size l is changed while the bit length of n is fixed at 1024. Similar to the blind computations, computational times remain largely unchanged for different l values.

An observation made through all our experiments is that the domain size l does not significantly affect the computational time of the key protocols Publish, Subscribe, Match and Cover, but the bit length n of the Paillier cryptosystem does. However, even for large bit lengths of n, our protocols take only a few microseconds or milliseconds and thus they are very efficient and practical.

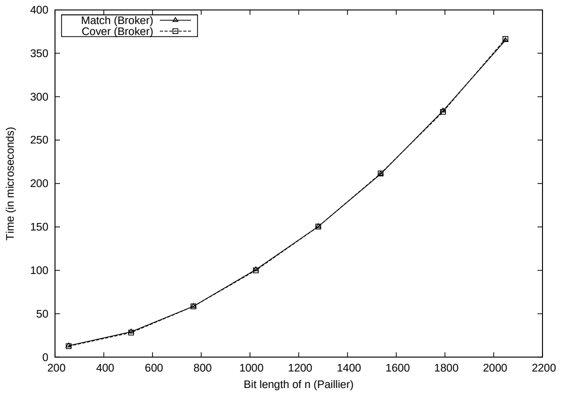

Fig. 7. Time to perform match and cover for different bit lengths of n

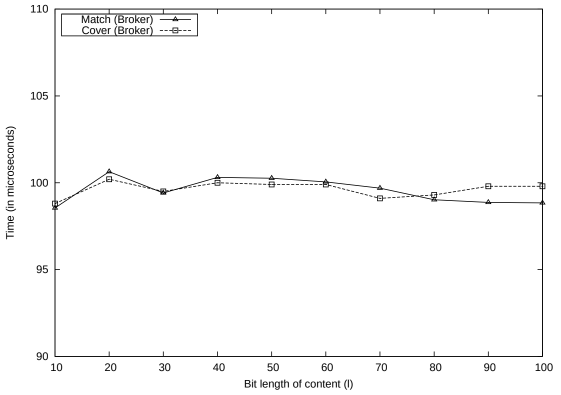

Fig. 8. Time to perform match and cover for different l

VIII. Conclusions and Future Work

We have presented an efficient cryptography-based approach to preserve subscription privacy and publication confidentiality in a CBPS system in which third-party Brokers perform `Match` and `Cover` protocols to make routing decisions for subscriptions without learning the actual content of the notifications published by Pubs and the subscriptions made by Subs. As described in Section VI, our protocols are secure and privacy preserving. The experimental results in Section VII show that the protocols are practical and efficient. Their executions take only a few milliseconds even for sufficiently large system parameters.

Managing subscriptions to effectively route notifications from Pubs to Sub through a large network of Brokers is a non-trivial task. There has been a considerable amount of research trying to address this problem in CBPS systems, without security and privacy issues being considered [10], [11], [4], [19], [20]. The privacy-preserving module our protocols create complements such research efforts, and can be used as a building block to design CBPS systems that efficiently route notifications while preserving the subscription privacy and publication confidentiality.

Our approach employs the Paillier homomorphic cryptosystem. In our future work, we plan to generalize the result by investigating the application of other additive homomorphic cryptosystems [14], [13]. We also plan to combine the current system with other techniques, including but not limited to, proxy re-encryption, searchable encryption, secure multiparty computation, and private information retrieval, to build a privacy-preserving CBPS system that can support a weaker trust model than we currently assume.

REFERENCES

[1] Bouncy Castle Crypto APIs. `http://www.bouncycastle.org/`.

[2] G. Ateniese, K. Benson, and S. Hohenberger. Key-private proxy re-encryption. In *CT-RSA '09: Proceedings of the The Cryptographers' Track at the RSA Conference 2009 on Topics in Cryptology*, pages 279–294, Berlin, Heidelberg, 2009. Springer-Verlag.

[3] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.*, 9(1):1–30, 2006.

[4] R. Baldoni, C. Marchetti, A. Virgillito, and R. Vitenberg. Content-based publish-subscribe over structured overlay networks. In *ICDCS 2005. Proceedings. 25th IEEE International Conference on Distributed Computing Systems, 2005.*, pages 437–446, June 2005.

[5] E. Bertino, B. Carminati, E. Ferrari, B. Thuraisingham, and A. Gupta. Selective and authentic third-party distribution of XML documents. *Knowledge and Data Engineering, IEEE Transactions on*, 16(10):1263–1278, Oct. 2004.

[6] M. Blanton. Online subscriptions with anonymous access. In *ASIACCS '08: Proceedings of the 2008 ACM symposium on Information, computer and communications security*, pages 217–227, New York, NY, USA, 2008. ACM.

[7] D. Boneh, G. Crescenzo, R. Ostrovsky, and G. Persiano. Public-key encryption with keyword search. In *EUROCRYPT 2004, Advances in Cryptology*, 2004.

[8] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. *Theory of Cryptography*, pages 535–554, May 2007.

[9] C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In *EUROCRYPT*, pages 402–414. Springer, 1999.

[10] F. Cao and J. Singh. Efficient event routing in content-based publish-subscribe service networks. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 929–940 vol.2, March 2004.

[11] A. Carzaniga, M. Rutherford, and A. Wolf. A routing scheme for content-based networking. In *INFOCOM*, pages 918–928, 2004.

[12] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, pages 41–50, Oct 1995.

[13] I. Damgård, M. Geisler, and M. Kroigard. Homomorphic encryption and secure comparison. *Int. J. Appl. Cryptol.*, 1(1):22–31, 2008.

[14] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *PKC '01: Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography*, pages 119–136, London, UK, 2001. Springer-Verlag.

[15] P. Eugster, P. Felber, R. Guerraoui, and A. Kermarrec. The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2):114–131, 2003.

[16] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *EUROCRYPT 2004, Advances in Cryptology*, 2004.

[17] C. Gentry and Z. Ramzan. Single-Database Private Information Retrieval with Constant Commun ication Rate. In *Proceedings of International Colloquium on Automata, Languages and Programming (ICALP)*, pages 803–815, 2005.

[18] E. Kushilevitz and R. Ostrovsky. Replication is NOT needed: Single database, computationally-private information retrieval. In *IEEE Symposium on Foundations of Computer Science*, pages 364–373, 1997.

[19] G. Li, S. Hou, and H.-A. Jacobsen. A unified approach to routing, covering and merging in publish/subscribe systems based on modified binary decision diagrams. *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on*, pages 447–457, June 2005.

[20] A. Machanavajjhala, E. Vee, M. Garofalakis, and J. Shanmugasundaram. Scalable ranked publish/subscribe. *Proc. VLDB Endow.*, 1(1):451–462, 2008.

[21] T. Matsuo. Proxy re-encryption systems for identity-based encryption. In *Pairing*, pages 247–267, 2007.

[22] K. Minami, A. J. Lee, M. Winslett, and N. Borisov. Secure aggregation in a publish-subscribe system. In *WPES '08: Proceedings of the 7th ACM workshop on Privacy in the electronic society*, pages 95–104, New York, NY, USA, 2008. ACM.

[23] S. K. Mishra and P. Sarkar. Symmetrically private information retrieval. In *Proc. of INDOCRYPT 2000*, pages 225–236, 2000.

[24] M. Nabeel and E. Bertino. Secure delta-publishing of XML content. In *ICDE, 2008. Proceedings of the IEEE 24th International Conference on Data Engineering*, pages 1361–1363, April 2008.

[25] K. Narayanam and C. Rangan. A Novel Scheme for Single Database Symmetric Private Information R etrieval. In *Proceedings of Annual Inter Research Institute Student Seminar in Computer Science (IRISS)*, pages 803–815, 2006.

[26] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology EUROCRYPT '99*, pages 223–238, 1999.

[27] T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO '91: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, pages 129–140, London, UK, 1992. Springer-Verlag.

[28] C. Raiciu and D. S. Rosenblum. Enabling confidentiality in content-based publish/subscribe infrastructures. In *Securecomm and Workshops, 2006*, pages 1–11, 28 2006-Sept. 1 2006.

[29] C. Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO '89: Proceedings on Advances in cryptology*, pages 239–252, New York, NY, USA, 1989. Springer-Verlag New York, Inc.

[30] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *SP ’00: Proceedings of the 2000 IEEE Symposium on Security and Privacy*, page 44, Washington, DC, USA, 2000. IEEE Computer Society.

[31] M. Srivatsa and L. Liu. Securing publish-subscribe overlay services with eventguard. In *CCS ’05: Proceedings of the 12th ACM conference on Computer and communications security*, pages 289–298, New York, NY, USA, 2005. ACM.

[32] S. G. Stubblebine, P. F. Syverson, and D. M. Goldschlag. Unlinkable serial transactions: protocols and applications. *ACM Trans. Inf. Syst. Secur.*, 2(4):354–389, 1999.

[33] C. W., A. Carzaniga, D. Evans, and A. Wolf. Security issues and requirements for internet-scale publish-subscribe systems. In *HICSS 2002. Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, pages 3940–3947, Jan. 2002.

[34] A. C. Yao. Protocols for secure computations. In *Foundations of Computer Science, 1982. SFCS ’08. 23rd Annual Symposium on*, pages 160–164, Nov. 1982.