P3S: A Privacy Preserving Publish-Subscribe Middleware

Partha Pal, Greg Lauer, Joud Khoury, Nick Hoff, Joe Loyall

P3S: A Privacy Preserving Publish- Subscribe Middleware

Partha Pal, Greg Lauer, Joud Khoury, Nick Hoff, Joe Loyall

BBN Technologies
Cambridge, MA 02138

{ppal,glauer,jkhoury,nhoff,jloyall}@bbn.com

Abstract. This paper presents P3S, a publish-subscribe middleware designed to protect the privacy of subscriber interest and confidentiality of published content. P3S combines recent advances in cryptography, specifically Ciphertext Policy Attribute Based Encryption (CP-ABE) and Predicate Based Encryption (PBE) with an innovative architecture to achieve the desired level of privacy. An initial P3S prototype has been implemented on top of a COTS JMS platform (ActiveMQ). Results of preliminary security analysis and initial evaluation of latency and throughput indicate that the P3S design is both practical and flexible to provide different levels of privacy for publish-subscribe messaging over various message sizes and network bandwidth settings.

Keywords: publish-subscribe, architecture, security, privacy, performance

1 Introduction

Message-oriented middleware supporting publish-subscribe (pub-sub) interaction has become fairly common in military and commercial applications. In pub-sub messaging, information consumers and producers do not need to establish a connection between them a-priori (often described as loose coupling). Pub-sub style messaging also provides selective filtering of information so that the consumers only receive messages they are interested in (often described as brokering). The combination of brokering and loose coupling facilitates scalability: instead of n entities each connecting to each other (n^2 connections), in a typical pub-sub system they just need to connect to the broker (n connections). However, loose coupling and brokering make it hard to maintain information privacy. Subscriber interest is usually visible at the broker because it needs to do the matching and filtering, and standard encryption cannot be used to protect the published content because there is no end to end security association between the information producer and the ultimate receiver of the content.

This drawback limits the use of pub-sub messaging in a wide range of system and application contexts. For example, in the commercial context, parties pursuing a merger and acquisition (M&A) deal may be interested in receiving updates on various topics, but the knowledge that party X is interested in topic Y may tip the hand of X.

In a military context, intelligence analysts in a coalition environment may be interested in receiving updates on information that they have agreed to share, but the knowledge that country A is interested in topic B may compromise country A's strategy. Also, in both the commercial and military contexts, information updates may have associated "need to know" type requirements stipulating that published content should not be visible to anyone other than the subscribers with matching interest—for example, the broker or other parties who are not interested in "Lehman Brothers" should not receive updated information about Lehman Brothers.

Techniques like sharing encryption keys among publishers and consumers, re-encryption, onion-routing etc. have been used to provide a level of privacy in pub-sub systems. We discuss a number of such approaches in Section 7; however, none of these provide a satisfactory solution to keeping subscriber interests private.

The main contributions of this paper are as follows:

- Design and implementation of a pub-sub middleware with a strong cryptographic guarantee of the privacy of subscriber interest and confidentiality of published content. To the best of our knowledge, no such system exists today.
- Performance analysis indicating that such privacy guarantee can be provided at a reasonable cost over a variety of combinations of message size, match rate and network bandwidth.
- Innovative combination of advanced cryptography with sophisticated architecture design as a blueprint for developing advanced security capabilities in the middleware.

The rest of the paper is organized as follows. Section 2 describes the privacy properties and performance characteristics that P3S set out to achieve. Section 3 presents the basics of the advanced cryptographic techniques used in P3S. Section 4 and 5 presents the P3S architecture design and current implementation respectively. Section 6 reports our preliminary analysis of privacy and performance. Section 7 summarizes related work, and Section 8 concludes the paper.

2 Terminology, and Privacy and Performance Targets

We use standard pub-sub terminology throughout the paper with a few exceptions. **Publisher** is an entity that wishes to make information content available to subscribers, and **subscriber** is an entity that registers subscription interest and receives the content that matches the interest. **Payload** refers to the content that a publisher wants to publish. The term **metadata** is used to refer to the description of a payload. **Interest** is a predicate about metadata and the term **matching** refers to the action of determining if the metadata describing a published payload satisfies the subscriber's interest. Among the less common terms, we use **third party** to denote an entity that is neither a publisher nor a subscriber. In P3S, there are four third parties namely, the Repository Server, the Distribution Service, the PBE Token Server and the Attribute-Based Access Control and Registration Authority (ARA). They will be introduced in

more detail in Section 4. Finally, we also use the term **participant** to mean a publisher, a subscriber or a third party.

The P3S middleware aims to satisfy a set of privacy and performance requirements above and beyond the traditional pub-sub functional requirements.

Basic P3S **functional requirements** are as follows. Publishers should not be aware of subscribers; P3S is expected to deliver published items to subscribers with matching interest. Matching is done based on metadata associated with published items, described as attribute-value pairs chosen from a fixed, predefined space of attributes and their values (metadata space). Subscriber interest is expressed as a conjunctive predicate over the attribute value pairs from the metadata space. The predicates may have wildcard (*) for values indicating interest in any value of the corresponding attribute. P3S should deliver a published item to a subscriber if and only if the latter is interested in the item. P3S should be open in the sense that legitimate clients may, within a metadata space, register any subscription. Controlling what subscription predicates a subscriber can issue is beyond the scope of the current paper; however we assume that a legitimate client behaving honestly will not subscribe with wildcards for all attributes.

The P3S **privacy requirements** are focused on protecting subscriber interest and minimizing the exposure of published content. Subscriber interests are private; other participants should not learn the interest(s) of a subscriber. Similarly, the publisher should not reveal payloads to subscribers unless their interests match. Furthermore subscribers should not learn anything about published metadata beyond knowing that their predicate does/doesn't match metadata. A publisher may not know if a particular item was matched or not. Other participants may learn that an item was published as long as the participant is unable to identify the item (for example the item is encrypted). An item that has been deleted based on its publisher's intent should not be available to subscribers even if the deleted item's metadata matched the subscriber's interest. Additionally, a participant with access to the original item may not re-publish the item after it has been deleted; he can however, publish the same content as a new item (new identifier, new metadata) as his own.

Finally, in terms of **performance requirements**, P3S aims to keep the average time to process and deliver a publication to an individual matching subscriber within ten times (10x) that of a similar (baseline) system without the privacy protection. Similarly, P3S throughput is also aimed to be no worse than ten times (10x) that of the throughput of a similar system without privacy protection.

3 Cryptographic Background

3.1 Predicate Based Encryption (PBE)

PBE is a 1: n (one-to-many) encryption scheme where encryption depends on the attribute values specified by the encryptor and where decryption keys depend on predicates. Decryption is possible only if the attribute values set by the encryptor (publisher in our case) satisfy the decryptor's (subscriber in our case) predicate. Formally, following the model and notation from [7, 6], let $\boldsymbol{x} = < x_1, x_2, \ldots, x_l >$ denote the

attribute vector with l elements chosen from the alphabet Σ, i.e., $\boldsymbol{x} \in \Sigma^l$, and let y denote the l-element interest vector chosen from the alphabet $\Sigma_* = \Sigma \cup \{*\}$, i.e., $\boldsymbol{y} \in \Sigma_*^l$ where $*$ denotes the wildcard character. Let us also define the conjunctive predicate $Match: \Sigma^l \times \Sigma_*^l \rightarrow \{0,1\}$ as $Match(\boldsymbol{x}, \boldsymbol{y}) = 1 \; when \; x_i = y_i \; \forall i \; for \; which \; y_i \neq *$. As in [7, 6], we focus only on the match predicate since it enables the construction of several other predicates.

Definition: A Public Key Predicate Based Encryption scheme (PK-PBE) consists of the following algorithms:

$Setup(\lambda) \rightarrow (PK, SK)$: The setup algorithm takes a security parameter and outputs a master public key PK and master secret key SK.

$Encrypt(PK, \boldsymbol{x}, M) \rightarrow CT_{\boldsymbol{x}}$: Encrypts the message M using the master public key PK and the attribute vector $\boldsymbol{x}$.

$GenToken(SK, \boldsymbol{y}) \rightarrow T_y$: Takes secret key $SK$and interest vector $\boldsymbol{y}$ and outputs a token $T_{\boldsymbol{y}}$.

$Query(T_y, CT_x) \rightarrow M$: Takes as input token $T_{\boldsymbol{y}}$ for some interest vector $\boldsymbol{y}$ and ciphertext $CT_{\boldsymbol{x}}$ encrypted using some attribute vector $\boldsymbol{x}$ and outputs message M if $Match(\boldsymbol{x}, \boldsymbol{y}) = 1$ and null otherwise.

Semantic security, token security, and collusion-resistance are the security properties generally considered in the context of a predicate encryption scheme. Semantic security requires that no information about the attribute vector $\boldsymbol{x}$ be revealed by the ciphertext. Token security requires that no information about the interest vector $\boldsymbol{y}$ be revealed by the cryptographic token. Collusion-resistance means that multiple tokens do not allow unauthorized decryption of a ciphertext, i.e., at least one of the tokens must match in order to decrypt, hence combining tokens does not release information. The public key encryption schemes in [6] and [7] provide semantic security and collusion-resistance and will be the focus of this work. The schemes do not provide token security. Any party with access to a token $T_{\boldsymbol{y}}$ and the ability to generate encrypted metadata is able to infer the interest vector $\boldsymbol{y}$ (see [9]).

Hidden Vector Encryption (HVE) [6] is an efficient PBE construction based on composite-order groups that assumes a single predicate – equality $Match$ - and supports large alphabets. Iovino et al. [7] provide a more efficient HVE construction that uses prime-order groups but restricts Σ to the binary alphabet $\Sigma = \{0,1\}$. The current implementation of P3S utilizes the construction and implementation in [7, 10]. While the P3S architecture is open to other PBE implementations, our choice of HVE implies that the supported predicates are conjunctions of equality on a binary alphabet augmented with wildcards. We extend the expressiveness of this binary PBE scheme to support a richer attribute space. Attribute and predicates are represented as bit vectors in HVE. To support a metadata space of N attributes, each of which may take one of 8 values, we construct the 3N-bit vector $\boldsymbol{x}$ where the first 3 bits are used to encode the 1st attribute, the next 3 for the 2nd and so on. We do the same for $\boldsymbol{y}$ and assume a wildcard spans all bits that represent the attribute. Note that the security of our mapping follows directly from the attribute-hiding property of HVE [7].

3.2 Ciphertext Policy Attribute Based Encryption

As in PBE, CP-ABE uses a set of attributes and a predicate over those attributes. However CP-ABE encrypts the ciphertext with a policy (predicate) over the set of attributes, and associates the decryption key with a set of attributes. Decryption of the ciphertext is possible if and only if the decryptor's attributes match under the predicate specified by the encryptor. CP-ABE is thus a 1: n (one-to-many) encryption scheme in which the encryptor of the data (publisher in our case) does not need to explicitly know who the participants (subscribers) are, yet can still constrain which participants may decrypt the data.

> ***Definition:*** A Ciphertext-Policy Attribute Based Encryption scheme (CP-ABE) [8] consists of the following algorithms:
> $Setup(\lambda) \rightarrow (PP, MSK)$: The setup algorithm takes a security parameter and outputs the public parameters $PP$and master key MSK.
> $Encrypt(PP, M, A) \rightarrow CT_A$: Encrypts the message Musing the public parameters PP and policy (also called the access structure) A defined over the attribute space. The algorithm outputs the ciphertext CT_A such that only a user that possesses a set of attributes that satisfy A is able to decrypt.
> $KeyGeneration(MSK, S) \rightarrow S_k$: Takes master key MSK and set of attributes $S$and outputs a private key S_k.
> $Decrypt(PP, S_k, CT_A) \rightarrow M$: Takes as input the public parameters and secret key S_k and ciphertext CT_A for some policy A. It outputs message M if the attributes satisfy the ciphertext policy.

In terms of its security properties, CP-ABE does not hide the policy$A$, or the attributes an entity holds. In fact, the policy$A$ is transmitted in the clear with the ciphertext. As with PBE, CP-ABE is collusion-resistant in the sense that combining keys can decrypt a message only if at least one of the keys can decrypt the message on its own.

We used the construction and implementation of Bethencourt et al [8, 15] in P3S. This construction does not support the logical operator NOT in A, a shortcoming that can be addressed by defining NOT of an attribute by a separate attribute, but this essentially doubles the number of attributes.

4 P3S Architecture

4.1 Components

The components of the P3S architecture are:

- Attribute-Based Access Control and Registration Authority (ARA): The ARA acts as the certification authority, and only interacts with other components during registration. During registration it provides the publishers and subscribers with information they need to publish, including the metadata and predicate schema, CP-ABE and PBE keying material (see Section 4.3 for more details).

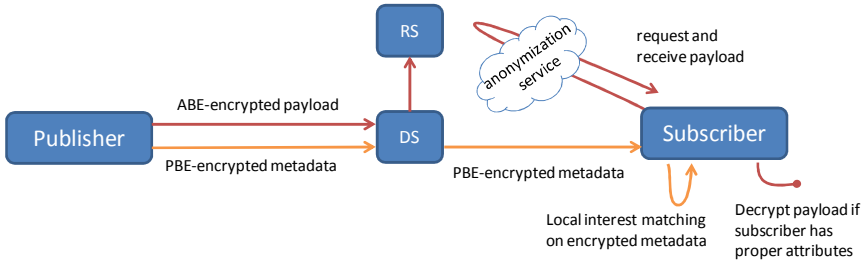
- Dissemination Server (DS): The DS sets up TLS tunnels to subscribers and publishers and keeps track of how to send information and acknowledgements to them. It receives PBE-encrypted metadata and CP-ABE-encrypted payload from the publishers, and forwards PBE-encrypted metadata to registered subscribers, and the CP-ABE-encrypted payload to the RS.
- Repository Server (RS): The RS stores CP-ABE encrypted payloads along with their associated Globally-Unique-IDs (GUIDs), and sends the encrypted payload associated with a GUID to a subscriber upon request.
- Predicate-Based Encryption Token Server (PBE-TS): The PBE-TS receives cleartext subscription interest (predicate) from the subscriber, and returns the corresponding PBE token to the subscriber.

The P3S architecture is designed to accommodate *anonymization*. If available, subscribers contact PBE-TS and RS via the anonymization service. P3S's basic privacy properties are independent of anonymization, but if incorporated, anonymization enhances privacy protection further by hiding the subscriber identity to PBE-TS and RS.

4.2 High Level Overview

Fig. 1 illustrates the basic high-level P3S information flow. Publishers use CP-ABE to encrypt payload with a policy that specifies what attributes are required to decrypt it. Subscribers have attributes that allow decryption of the CP-ABE encrypted payload if they satisfy the publisher's CP-ABE policy. In this sense, CP-ABE provides a level of access control to protect the confidentiality of the payloads. Subscribers obtain PBE tokens representing their subscription predicates. Publishers PBE encrypt a reference to the payload using the associated metadata and send the encrypted metadata to subscribers, via the DS. Subscribers match their tokens against the encrypted metadata. A successful match yields the only information required to retrieve the payload from the RS. Performing the matching in the subscriber combined with the use of PBE protects the privacy of both subscriber interest and content metadata. The retrieval request is then sent through an anonymization service (if available).

Fig. 1. P3S high level architecture



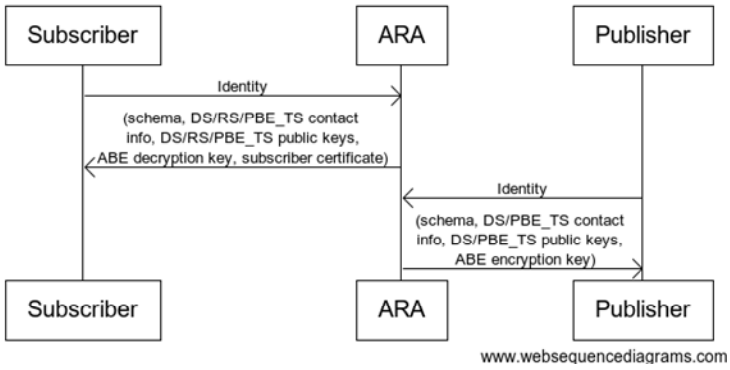
The CP-ABE encryption allows the publisher to control who can see the payload without requiring the publisher to know which subscriber is receiving it. The PBE encryption allows the subscriber to determine which publications match its interests

without the system disclosing the metadata associated with the publication. Recall that the access policy in CP-ABE encryption is "in the clear", and thus the access policies should only refer to attributes that are safe to disclose to subscribers that may fail to decrypt the payload, such as organization names or subscriber roles. PBE encryption does not disclose the values of the attributes used to encrypt the data (except to the extent that a match with subscriber predicates discloses it). However, our current architecture does not provide a mechanism to restrict the types of queries that a subscriber can make. The next section discusses the P3S protocol operation in detail.

4.3 Operation

Initialization: Fig. 2 illustrates the initialization process for Subscribers and Publishers. The ARA provides the subscriber with the PBE metadata format, i.e., field/value information for specifying subscription interests, contact information for the P3S services (RS, DS and PBE-TS) and their public key certificates, a CP-ABE secret key (SKC) based on the client attributes, which is used to decrypt payloads, and a certificate that indicates the participant is a subscriber.

Fig. 2. P3S initialization process

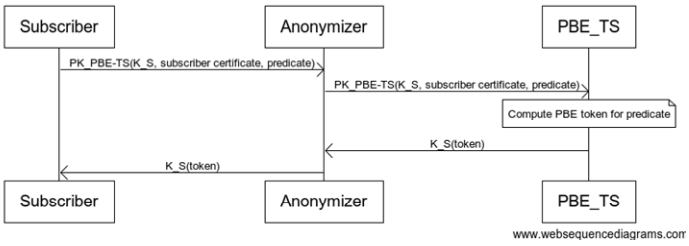


The ARA provides the publisher with the PBE public parameters, metadata format, contact information and public key certificates for the P3S services (DS and PBE-TS), and the CP-ABE policy attributes and the CP-ABE public parameter PKC to be used by the publisher to encrypt the contents it wishes to publish.

Subscription: Fig. 3 illustrates the process of subscription. The subscriber generates a symmetric key K_S and then uses the public key of the PBE-TS to encrypt the 3-tuple $(K_S,\ subscriber\ certificate,\ plaintext\ predicate)$ and sends it to the PBE-TS via the anonymization service. The PBE-TS decrypts the triple and, if the subscriber certificate is valid, computes the PBE token corresponding to the plaintext predicate. It then encrypts the token using the key K_S and sends it back to the (unknown) subscriber via the anonymization service. This process allows the subscriber to obtain the

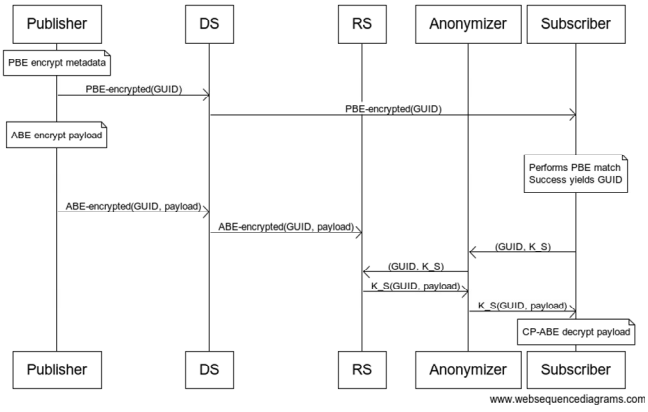
token associated with its plaintext predicate while remaining anonymous to the PBE-TS providing the token. Note that the PBE-TS sees the plaintext predicate.

Fig. 3. P3S subscription process



Publication: Fig. 4 illustrates the process of publication. The Publisher has a payload and associated metadata to be published. It generates a unique $GUID$ from a large space (making it hard to guess) and then uses PBE encryption to encrypt that $GUID$. It sends this PBE-encrypted $GUID$ to the DS which then forwards it to all the subscribers. The Publisher then CP-ABE encrypts the 2-tuple $(GUID, Payload)$. The CP-ABE encryption specifies a policy that defines the attributes required by a subscriber if the payload is to be decrypted. The choice of this policy is outside of the scope of this paper, but could be determined by the payload metadata. The Publisher then sends the 3-tuple $(GUID, CP\ ABE\ encrypted(GUID, Payload), T_{GUID})$, where T_{GUID} represents a time to live (TTL) for this item, to the DS which forwards it to the RS. The RS stores the $CP\ ABE\ encrypted(GUID, Payload)$ indexed by the GUID for later retrieval by subscribers. The RS stores the item for at least T_{GUID}, after which it is garbage collected.

Fig. 4. P3S publication process



The subscribers receive the PBE-encrypted GUID from the DS and attempt to decrypt it using their PBE tokens. If a subscriber's predicate matches the metadata used during the PBE encryption of the GUID, the GUID will be revealed. The subscriber is then ready to request the associated payload. It first generates a symmetric key K_S and then encrypts the 2-tuple $(K_S, GUID)$ with the RS's public key. It then sends this message to the RS via an anonymization service. The RS decrypts the message, retrieves the $CP\ ABE\ encrypted(GUID, Payload)$ associated with the GUID, encrypts that $CP\ ABE\ encrypted(GUID, Payload)$ using the key K_S and sends it back to the subscriber via the anonymization service. This process allows the subscriber to request a particular payload without revealing its identity to the RS.

The subscriber decodes the message using the key K_S and then attempts to CP-ABE decode the $CP\ ABE\ encrypted(GUID, Payload)$. If it has the right attributes to successfully decode it then it obtains the payload and associated GUID (which it then uses to correlate the request and response).

Deletion: Deletion is handled by the RS's garbage collection mechanism. The RS has a configurable parameter T_G and each publication provides the item specific TTL T_{GUID} which represents the publisher's intent to delete the content after the specific period of time. The RS deletes the item corresponding to the identifier $GUID$ after $T_{GUID} + T_G$. The reason for the configurable T_G parameter is to provide some accommodation for the uncontrollable delays in a distributed setting and slower consumers. For a strict interpretation of deleting based on publisher's intent T_G can be set to 0, which may result in considerably more failures to fetch the item for some (slower) clients with matched subscription.

5 Current Prototype

We have implemented a P3S prototype using the Apache Active MQ [14] open source Java implementation of the Java Message Service (JMS) standard. The current version of the prototype includes all components and features described above except for the anonymization service and CP-ABE encryption (i.e, the ARA and encryption of published content). We have developed the CP-ABE support functionality using the construction and library described in [8, 15], but it is not yet integrated with the P3S prototype. The PBE encryption support is implemented by enhancing the HVE implementation [7] of the JPBC [10] library, and is integrated in the prototype.

In the current implementation, the DS is implemented by extending the AMQ broker. The P3S subscriber and publisher protocols are implemented by extending the AMQ client libraries. We retained the top level JMS interface, so that existing JMS compliant publishers and subscribers can take advantage of the P3S's privacy preserving properties without code change, once they include the P3S enhanced AMQ client libraries. The RS is implemented as a composition of two services, a Web Service to respond to subscriber's request for content retrieval, and a Persistence Service that subscribes to the DS for encrypted content and uses an embedded Apache Derby database for storing them. The PBE-TS is also implemented as a Web Service that runs within an embedded Jetty container and embeds the extended HVE library. We plan

to implement the ARA in a similar manner. Publishers and subscribers interact with the DS over TLS.

6 Analysis of Privacy and Performance Overhead

6.1 Privacy

In this paper we present a semi-formal analysis of privacy. We begin with the threat model considered in the analysis:

> ***Definition:*** An *Honest but Curious* (HBC) participant only makes well-intentioned requests (honest) but remembers everything that was sent to them (curious). They do not eavesdrop, masquerade as other participants, or hijack communications.
>
> ***Definition:*** A *malicious* participant attempts to eavesdrop, performs replay and man-in-the-middle attacks, and masquerades as other participants.

Note that colluding HBC participants may share information without being malicious. Our analysis focuses mainly on privacy under an HBC threat model, but includes colluding HBC subscribers. The ARA, which we assume to be a trusted certification authority, is not part of the analysis. Additionally, integrity and availability are also kept out of scope for the most part except for the following. Because of TLS and the request-response nature of P3S messages, participants can detect if network failures cause message loss at the application level. The basic P3S operation is robust to node failures as well. The RS stores encrypted content on disk. A crashed component can resume publish-subscribe activities after restart without requiring re-encryption of any published content. A restarted subscriber simply needs to (re)register with the DS and (re)obtain its PBE tokens from the PBE-TS. Similarly, upon restart a publisher needs only to (re)register with the DS. A restarted DS needs to wait for subscribers and publishers to (re)register. A restarted RS simply needs to (re)register with the DS.

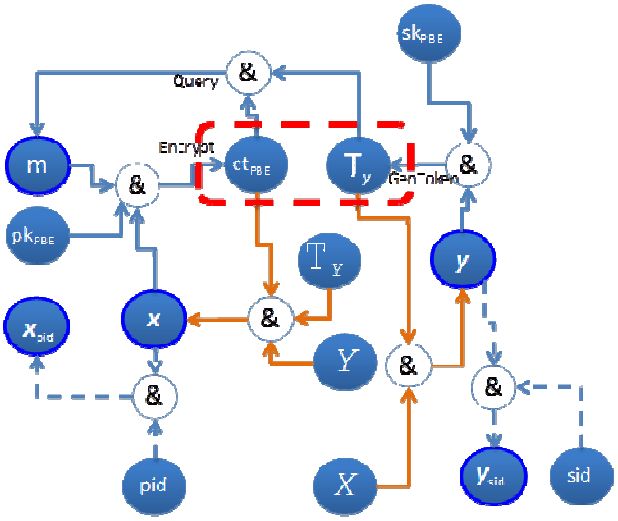
Structured Analysis Using *Gadgets:* A *gadget* is a simple mechanism we developed to capture information dependency underneath an encryption scheme. In this section we use the PBE gadget that captures PBE information elements and their interdependencies as an illustration. Gadgets for other encryption schemes used (e.g., CP-ABE, Public Key, Symmetric Key) in P3S are similarly constructed.

More specifically, a gadget is a directed graph $G = (V, E)$ where each node in V is either an information element or an AND gate (&). Nodes in the red boundary are the main information elements, the ciphertext and the token to unlock it. Edges in E represent information dependencies: a directed edge from node u to node v means that information element v depends on u. When u is the & gate, then v depends on all information elements that are incident to u.

Fig. 5 shows the PBE gadget. Upper case labels like X, Y, T_Y represent the set of all possible information elements represented by their lower case counterparts $\boldsymbol{x}, \boldsymbol{y}, T_{\boldsymbol{y}}$. In **Fig. 5**, PBE ciphertext ct_{PBE} depends on the (plaintext) message m (which in P3S is a GUID) and the attribute vector $\boldsymbol{x}$ (which in P3S is the metadata description) and the PBE master public key pk_{PBE}. The & gate leading to ct_{PBE} embodies the PBE En-

crypt operation described in section 3.1. Similarly, information elements and dependencies underneath the two other major PBE operations, namely GenToken (which takes the interest vector $\boldsymbol{y}$ and the PBE master secret key sk_{PBE} to produce the PBE token T_y corresponding to $\boldsymbol{y}$) and Query (which takes the ciphertext ct_{PBE} and PBE token T_y to recover m) are also shown in the PBE gadget. A gadget can be extended to represent additional dependencies relevant to the system using the encryption scheme represented by the gadget. For instance, in **Fig. 5** nodes connected only by broken edges are additional information and dependency that matters to P3S such as the association $\boldsymbol{x}_{pid}$ between the publisher identity pid and the attribute vector $\boldsymbol{x}$ (representing metadata), and the association $\boldsymbol{y}_{sid}$between subscriber identity sid and the interest vector $\boldsymbol{y}$ (representing subscriber interest). Nodes in the extended gadget with dark borders represent the information subject to privacy requirements.

Fig. 5. PBE Gadget



Analysis using the PBE gadget described above involves tracing the execution steps of the P3S system over time focusing on the behavior of individual participants and information they become privy to during execution. We then test whether private information (information elements with dark borders) becomes visible to undesired participants, and if so, under what circumstances (i.e., HBC, malicious or colluding). Undesired exposure of sensitive information *M* is a threat to the privacy of *M*.

In any execution of P3S, the PBE-TS can see all subscription interests in plain text. However, because of the anonymizer, the PBE TS cannot associate the subscription interests to subscriber identities. Privacy of $\boldsymbol{y}_{sid}$ is still maintained even though the interest vector $\boldsymbol{y}$ is visible to the PBE-TS. Similarly, anyone who has access to the ciphertext ct_{PBE} and the right PBE token can decrypt it. However, under HBC operation, a subscriber's token is not shared with anyone else, and the PB-TS does not see ciphertext ct_{PBE}. Therefore privacy of m is not threatened.

Analysis using the PBE gadget illustrates the lack of token security in PBE [9]. If a participant is able to obtain a token $T_{\boldsymbol{y}}$ and create encrypted metadata, it will be able to reveal $\boldsymbol{y}$ by creating encrypted metadata for all attribute vectors (i.e., X) and test them against the token $T_{\boldsymbol{y}}$**.** This threat is indicated by the orange edges connecting the & node with $\boldsymbol{y}$, X and $T_{\boldsymbol{y}}$. In HBC execution of P3S, all non-3rd party participants can encrypt any attribute vector in X, but they only have access to their own tokens. A colluding HBC subscriber S_1 can share its Token $T_{\boldsymbol{y}}$with others, but if they can do that, they might share their plaintext interest as well. Even then, such sharing does not reveal any more information than the union of the information revealed by them individually. A malicious non-3rd party participant however can obtain any token $T_{\boldsymbol{y}}$, i.e., privacy of $\boldsymbol{y}$ (subscriber interest) is threatened under malicious participants.

Another issue revealed by the gadget is that if a subscriber can subscribe to all or a significant part of the space of all possible subscription interests (i.e., Y) to accumulate $T_{\boldsymbol{Y}}$, he can test any given ciphertext ct_{PBE} against all tokens in $T_{\boldsymbol{Y}}$ to reveal the attribute vector $\boldsymbol{x}$ used to encrypt ct_{PBE}. This is shown by the orange edges connecting the & node with $\boldsymbol{x}$, Y , ct_{PBE}, and T_Y. HBC and non-colluding execution of P3S will not allow a subscriber to share tokens, however, over time a subscriber might accumulate a large number of tokens, which could be used to launch this attack. We have identified ways to mitigate this threat. One possibility is to time-stamp publications and tokens, making tokens active only within a configurable period of time. This approach has the advantage of providing a token revocation mechanism but requires the clients to be time-synchronized and using time as an additional metadata attribute.

Summary of non-3rd party participant's visibility: An HBC subscriber does not know about anyone else's subscription interest. It does not know metadata description of published payloads even though it receives all PBE encrypted metadata. PBE matching, even when the match succeeds, does not reveal the metadata description. Matched metadata reveals the GUID, but the subscriber cannot see the corresponding content unless it possesses the appropriate CP-ABE attributes to decrypt the CP-ABE encrypted $(GUID, Payload)$ pair. Being able to decrypt the payload does not reveal the publisher identity unless the identity is included in the content. A subscriber that also publishes of course has full visibility of its publications (content and metadata).

An HBC publisher will have no visibility of content and metadata being published and subscribed by other participants. The publisher does not know whether the content it published matched with anybody's subscription, or the identity of the matching subscriber, or whether anyone actually received its content.

Summary of 3rd party participant's visibility: The HBC RS does not know which publisher has published, since it receives all messages from the DS. It does not know the content of the message since they are CP-ABE encrypted, and as a result does not know anything about the content of the payload it sends to a subscriber. It does not know the metadata associated with the content since that information is PBE encrypted and not delivered to RS. The RS does not know which subscriber has requested a payload, since all such requests are received from an anonymization service. The symmetric key K_S sent with such requests allows the RS to return the payload to the subscriber privately without having to know the subscriber's identity. The RS can

keep track of whether a CP-ABE encrypted payload has ever been requested and how many requests have been received for each such encrypted payload. It knows neither the plaintext payload nor the metadata associated with an encrypted payload.

The HBC DS knows nothing about the subscriber interests since those are kept local to the subscribers. The DS does not know the content of the payload, since it is CP-ABE encrypted. It does not know anything about the metadata associated with a payload since that information is PBE encrypted. The DS does not know which payloads have been requested since it does not see any requests for payload from subscribers and, in any event, such requests are encrypted with the RS's public key. The DS does know the size of payloads and the size of encrypted PBE metadata.

The HBC PBE-TS does not know anything about publications as it receives no encrypted metadata and no encrypted payloads. The PBE-TS knows the plaintext predicates generated by subscribers but does not know the binding of subscriber to predicate as all PBE token requests are sent via the anonymization service. The symmetric key K_S sent with such requests allows the PBE-TS to return the token to the Subscriber privately without having to know the Subscriber's identity.

Eavesdroppers and Other Leakage: Eavesdroppers without any CP-ABE or PBE credentials learn nothing about subscriptions, metadata or payload content. Eavesdroppers may learn the GUID sent by the publisher in the clear but may not decrypt the associated CP-ABE payloads[1]. To prevent eavesdroppers from learning if more than one subscriber has received the same payload, transmissions of a payload from the RS to subscribers are super-encrypted with a subscriber-specified symmetric key. Requests for payloads are encrypted with the RS's public key. Legitimate interactions in P3S however reveal a number of auxiliary information about P3S to parties that are not the intended receiver of such information. For example, the size of encrypted content (subscribers and RS are legitimate end users of this interaction) is visible to eavesdroppers as well as the DS. CP-ABE access control policy is visible to the RS (matching subscribers are legitimate end users of this interaction). The RS knows if an encrypted content has been sent to some subscriber(s) (i.e., matched). The aggregate rate at which items are being published can be estimated by subscribers from the number of encrypted metadata they're getting. The RS can estimate it by how frequently payloads are stored. Eavesdroppers and the DS know the per-publisher publication rate and number of items published by each publisher. Eavesdroppers and the RS know the aggregate number of items received by subscribing clients.

6.2 Performance

We collected metrics by running the P3S prototype in various configurations such as all parties on one physical server, the DS and RS on a server and a small number of other participants on individual hosts in the network. However, these measurements do not present the true performance characteristics of P3S. Even though only a fraction of the subscriptions may actually match a given publication, it is important to

[1] To protect against this the publisher may super-encrypt the GUID with the RS's public key before publishing the payload message.

consider all subscribers in the model because the baseline needs to test each subscription against a publication (impacts the broker's processing load), and encrypted metadata for each publication needs to be disseminated to all clients (consumes network bandwidth). Therefore, we used analytic models with parameter values obtained from the current prototype to get an understanding of the performance at scale (e.g., 100s of subscribers) of the P3S system vis-à-vis a baseline. We used a standard centralized pub-sub system as baseline, where publishers submit their payload and metadata (such as a topic) to a central broker, subscribers register subscriptions with the broker, and the broker sends the payload whose metadata matches with a subscription to the subscriber. In the P3S model, we ignored the anonymizer since as explained in the previous section, anonymization is not necessary for the basic privacy guarantees of P3S. Since CP-ABE is not yet integrated in the P3S prototype, we obtained the CP-ABE timing and ciphertext sizes from the CP-ABE library running standalone. **Table 1** shows the parameters of the model and their values used in the analysis. The two metrics we evaluated are end-to-end latency and throughput:

- **End-to-end Latency**: This is the time taken by a single publication to reach all matching subscribers including the time taken for encryption and decryption.
- **Throughput**: This is the maximum rate at which publications can be injected into the system, such that all are properly matched and delivered.

Table 1. Parameters and values used in performance models

Symbol	*Meaning*	*Input Values*
ℓ	Network latency	45 ms
$\mathcal{B}$	Network bandwidth	10 Mbps
M	Size of plaintext payload to be transferred	Varying
P	Size of PBE metadata specification	40 bits
P_P	Size of PBE-encrypted metadata	10KB
M_A	Size of CP-ABE-encrypted payload	$0.6 \times M$
$ser(m)$	Serialization time for message size m	$m/\mathcal{B}$
N_s	Number of subscribers	100
f	Fraction of subscribers that match a given publication	5%
V	Number of attributes in CP-ABE policy	10
K	Security parameter in CP-ABE algorithm	384 bits

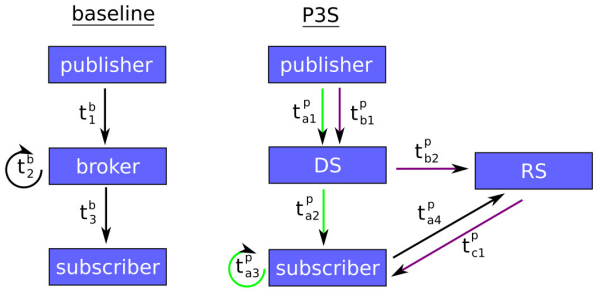
Sketch of the End-to-end Latency Model

The major contributors to end-to-end latency are shown in **Fig. 6**. End-to-end latency for the **baseline** $t^b = t_1^b + t_2^b + t_3^b$, where t_1^b is the time for the publisher to send its message (with metadata) to the broker, t_2^b is the time for the broker to perform the matching operation against all registered subscriptions, and t_3^b is the time for the broker to transmit the message to **all** matching subscribers.

Messages from one node to another incurs a fixed latency ℓ and a serialization time $ser(m)$, where m is the message size. Given a network bandwidth $\mathcal{B}$, $ser(m) =$

$m/\mathcal{B}$. The baseline system may use standard cryptography (e.g., SSL) to encrypt messages, but difference in the size of cleartext and the corresponding ciphertext is insignificant to impact the processing and transmission times, which means $t_1^b = \ell + \frac{M}{\mathcal{B}}$. Because the broker needs to send published item to $f \times N_s$ matching subscribers, and sending an item to a single subscriber takes the same time as t_1^bs, $t_3^b = f \times N_s \times t_1^b$. Simple XPath matching operation in a modern desktop takes roughly .05ms, and therefore with N_s subscribers $t_2^b = 0.05 \times N_s\, ms$.

Fig. 6. Contributors to latency for baseline and P3S



End-to-end latency for **P3S** $t^p = \max(t_a^p, t_b^p) + t_c^p$, where t_a^p is the elapsed time between the publisher sending the encrypted metadata to the time when the RS receives the content retrieval request from the last matching subscriber, t_b^p is the time to send the encrypted item to RS (i.e., content submission), and t_c^p is the time taken for RS to send the encrypted item to requesting subscribers. As shown in **Fig. 6** t_a^p has four subcomponents ($t_{a1}^p, t_{a2}^p, t_{a3}^p, t_{a4}^p$), and and t_b^p has two (t_{b1}^p, t_{b2}^p). Activities contributing to t_a^p and t_b^pcan happen in parallel in P3S, but until both complete, the RS cannot serve the requested item, hence we take $\max(t_a^p, t_b^p)$. This formulation of t^p is actually a worst case estimate, which happens when matching subscribers receive the encrypted metadata last, and the last matching subscriber requests the content first. In practice, activities contributing to t_c^p happen in parallel to t_a^p or t_b^p because a subscriber requests the content as soon as its subscription matches, and the RS may receive the item while some subscribers are still receiving the encrypted metadata.

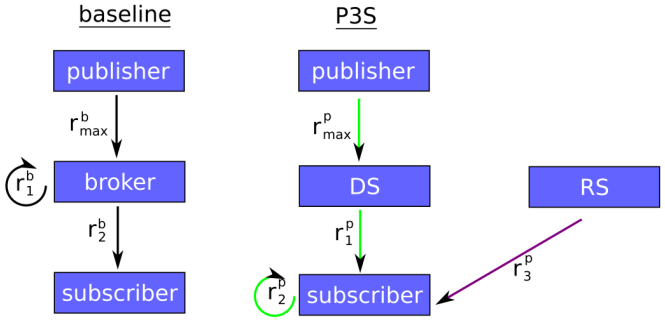
The time taken to PBE encrypt and send the metadata to the DS (t_{a1}^p) is $\ell + ser(P_P) + enc_P$, where enc_P is the PBE encryption time($\approx 30ms$ in our test environment). The time from when the DS starts dissemination to when the last subscriber receives the transmission (t_{a2}^p) is $\ell + N_s ser(P_P)$. The time to perform PBE match operation (t_{a3}^p) is $\sim 30ms$ for the ~40KB ciphertext. The last component of t_a^p, $t_{a4}^p = \ell + ser(G)$ where G denotes the size of a GUID, which is ~10 bytes. The component t_{b1}^p includes the time to CP-ABE-encrypt the content and send it to the DS, and is approximated as $\ell + ser(M_A) + enc_A$. CP-ABE encryption is fairly fast ($enc_A \approx 3ms$). The component t_{b2}^p is also estimated to be $\ell + ser(M_A)$ but the serialization times will be different because the bandwidth between the DS and RS is assumed to be 100 Mbps, as might be typical on a LAN, whereas the bandwidth be-

tween the publisher and the DS is 10Mbps as shown in **Table 1**. CP-ABE encryption adds to the size of the message being sent, the size of the CP-ABE ciphertext is estimated from theory to be $M_A = 2VK + M$, where V is the number of attributes in the CP-ABE policy, K is a security parameter, and M is the size of the plaintext payload. Finally, the component t_c^p, corresponding to the time taken to send the payload from the RS to all matching subscribers and subsequent decryption, can be modeled similarly to the dissemination of PBE-encrypted metadata i.e., $t_c^p = \ell + ser(M_A) \times N_S \times f + dec_A$. The last subscriber to get the payload has to wait for the serialization time for all other subscribers, the serialization time for itself, the network latency for itself, and the time to do the CP-ABE decryption ($dec_A \approx 12ms$).

Sketch of the Throughput Model

Fig. 7 shows the major contributors to our throughput model. We determine the maximum rate at which each part of the system can process publications, and find the minimum of those rates. For the baseline, the throughput is min (r_1^b, r_2^b) where r_1^b is the rate at which the broker can match publications to subscriptions, and r_2^b is the rate at which the broker can send payloads to matching subscribers. Given β hardware threads for matching, $r_1^b = \frac{\beta}{N_S \times t_{xpath}}$, and $r_2^b = \frac{\beta}{M \times N_S \times f}$.

Fig. 7. Major contributors to throughput for baseline and P3S



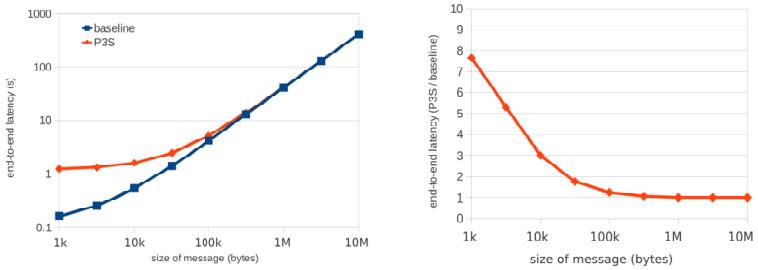
For P3S, the throughput for P3S is min(r_1^p, r_2^p, r_3^p), where r_1^p is the rate at which the DS can broadcast metadata items to subscribers and modeled as $r_p^1 = \frac{\mathcal{B}}{P_P \times N_S}$; r_2^p is the rate at which a subscriber can perform PBE matches, and modeled as $r_p^2 = \frac{\alpha}{t_{PBE}}$, where t_{PBE} be the time required for the subscriber to match their PBE token and α is the number of hardware threads dedicated to matching (currently set to 2); and r_3^p is the rate at which the RS can satisfy payload requests, and modeled as $r_3^p = \frac{\mathcal{B}}{M_A \times N_S \times f}$. The subscriber sends a request to the RS when a match occurs, these requests are small and infrequent, and are not a limiting factor.

Results

In this section, the message size in the horizontal axis of all figures refers to clear text payload size. **Fig. 8** shows the latency results for $\mathcal{B} = 10Mbps$. For small payloads,

the baseline has low latency (**Fig. 8(a)**) because it only has to serialize a small number of small messages, and matching time is small. As the payloads get larger, latency is dominated by the serialization time in the available bandwidth. The P3S system follows the baseline for large payloads. For large payloads, network serialization time dominates over other factors such as the PBE matching time. For small payloads P3S exhibits a threshold. The PBE matching operation at the subscriber takes approximately 38ms, regardless of clear text payload size. For 1K payloads, all network operations take 1ms or less. Accordingly, for small payloads, the P3S system performance is within ten times the baseline (**Fig. 8(b)**).

Fig. 8. End to end latency analysis

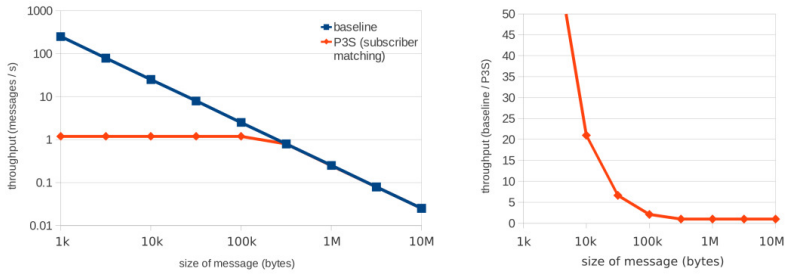


(a) end-to-end latency over varying message size

(b) latency relative to baseline

Results for the throughput analysis are shown in **Fig. 9**. As with latency, bandwidth is the dominant factor in the baseline. As payload size increases, throughput decreases because fewer messages per second can be sent out the network interface of the central broker. The P3S system exhibits almost exactly the same behavior as the baseline for large payloads, but it is the bandwidth out of the RS that limits the throughput.

Fig. 9. Throughput analysis for f = 5%



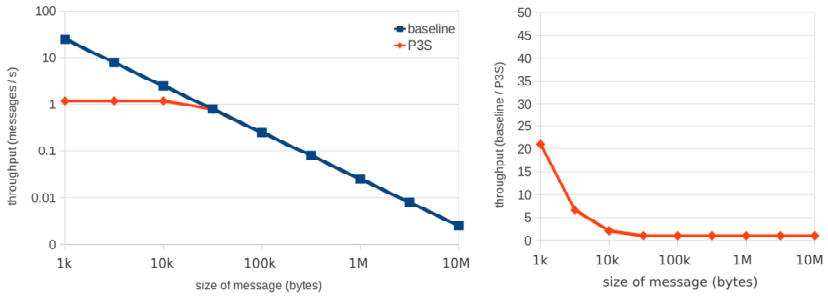
(a) throughput over varying message size

(b) throughput relative to baseline

For small payloads, P3S performance flattens because regardless of the payload size, the DS must send the PBE encrypted metadata (~40KB) to each of the 100 subscribers, which creates a bottleneck in the network interface of the DS. Consequently, P3S performs worse than the baseline for small payloads (**Fig. 9(b)**). This issue can be addressed by reconfiguring the P3S architecture to use hierarchical dissemination.

P3S throughput relative to the baseline shows no dependence on the number of subscribers for a fixed matching rate *f*. We also observed that increasing the network bandwidth from 10 to 100 Mbps helps both systems equally. But increasing the match rate benefits P3S. The baseline only disseminates to subscribers who match, whereas P3S must disseminate to all of them, and if more subscribers match, the baseline loses its advantage. **Fig. 10** shows the throughput of both P3s and the baseline for $f = 50$. By contrast, the plots in **Fig. 9** was for $f = 5\%$. Combining all these results, we can conclude that P3S performs very well (within 10x) compared to the baseline except for small payloads and low matching rates.

Fig. 10. Throughput analysis, f = 50%



(a) throughput over varying message size

(b) throughput relative to baseline

7 Related Work

Standard security measures such as role-based access control and content encryption in traditional pub-sub middleware offer only a partial solution: the decryption key needs to be shared among potential subscribers and enough metadata and subscription information needs to be visible to the broker. A content-based pub-sub scheme where content decryption keys are shared using Pedersen commitment and matching is performed on blinded attribute-value pairs is presented in [5]. Although their scheme has similar objectives, subscribers need to register a-priori with the publishers, and brokering is limited to equality of strings and numeric comparison. Another approach outlined in [4] makes use of reencryption and onion-routing indirection to dissociate the location of predicate matching the publishers and subscribers. This scheme appears to be specialized for a P2P content sharing network. In [3] a policy-based approach is presented where data owners can specify who can access their publications and under what condition. But the broker and the policy enforcement mechanism can see both the published content and subscriber interest. Contrail [16] presents a novel form of pub-sub for smart phones that uses sender-side content filters for privacy. In this scheme, the association between the publisher and subscriber is pretty strong- the subscriber and publisher perform a handshake to install the sender-side filter. Private stream searching [17] is another relevant research area where the goal is to run encrypted query on unencrypted streams to produce encrypted matching results.

Homomorphic encryption [12] offers a potential solution for privacy preserving pub-sub however, homomorphic encryption supporting complex computation performed at the broker is still not practical. We are not aware of any work attempting to preserve the privacy of subscriber interest and confidentiality of published content in the way described here other than the two other projects under the R&D program supporting this work. One uses circuit-based minimal model of secure computation [2] and Barrington's theorem [1] to simulate the complexity class NC^1 using width-5 branching programs. The other uses Oblivious Transfer [13] to achieve the privacy objectives.

8 Conclusion

Current pub-sub systems do not provide privacy of published metadata or subscriber's interest, and can only provide a limited cover for published content. The P3S system is designed to protect the privacy of subscriber interest and confidentiality of published content. A P3S prototype is implemented on a COTS JMS platform (Apache AMQ). The privacy guarantees of P3S come from innovative use of PBE and CP-ABE, and an innovative system architecture that severely limits the exposure of private information by isolating and careful positioning of key underlying information and computation. All components required to support P3S protocol interactions have been developed, the initial integrated P3S prototype integrates all capabilities except for CP-ABE encryption and anonymization of subscriber interactions with the PBE-TS and RS. Initial evaluation shows that P3S overhead is within 10x of the baseline for a variety of payload size, match rate and network bandwidth combinations. Preliminary privacy analysis shows that P3S preserves the privacy of published content and subscriber interest for HBC participants, and even when some of them collude.

Analysis also revealed a number of shortcomings of the current P3S prototype. For example, the PBE-TS is privy to plaintext subscriber interest. Also, there is no subscription control policy enforced on the subscribers. We are currently investigating how to address these shortcomings. One potential approach is to find alternative configurations where subscriber interest never gets out of the subscriber. For instance, the PBE-TS functionality can be embedded in each subscriber instead of being centralized. Another alternative is to frame PBE Token generation as a secure 2-party computation [11] in which the PBE TS has the PBE Master Key, and the subscriber has the interest, and a PBE token is produced by a secure 2-party computation between them without divulging any party's information to the other. Apart from these, we are also exploring innovative uses of the basic privacy-preserving pub-sub middleware such as private multiparty chat or private control channels in a control system. Finally, we are performing formal security analysis of P3S using indistinguishability games to complement the semi-formal analysis presented in this paper.

9 Acknowledgement

The authors acknowledge the collaboration and guidance of Brent Waters and Vitaly Shmatikov of Univeristy of Texas, Austin in integrating CP-ABE and PBE in P3S.

This work is supported by the Intelligence Advanced Research Project Activity (IARPA) via Department of Interior National Business Center (DoI/NBC) Contract No. DIIPC20195. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DOI/NBC, or the U.S. Government.

10 Bibliography

1. D. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1, *Journal of Computer and System Sciences* **38:1** (1989), 150-164.
2. U. Feige, J. Kilian, M. Naor. A minimal model for secure computation (extended abstract). STOC 1994: 554-563.
3. L. Opyrchal, A. Prakash, A. Agrawal. Supporting Privacy Policies in a Publish-Subscribe Substrate for Pervasive Environments. *Journal of Networks*, North America, 2, Feb. 2007.
4. M. Klonowski, M. Kutylowski, B. Rozanski. Privacy Protection for P2P Publish-Subscribe Networks, Security and Protection of Information, Brno Univ. of Defense 2005, 63-74.
5. M. Nabeel, N. Shang, E. Bertino. Privacy-Preserving Filtering and Covering in Content-Based Publish Subscribe Systems. Purdue University Tech Report 2009-15, 18 June 2009
6. D. Boneh, B. Waters. Conjunctive, Subset and Range Queries on Encrypted Data. Proceedings of the 4th Conference on Theory of Cryptography, Amsterdam, 2007, 536-554.
7. V. Iovino, G. Persiano. Hidden-Vector Encryption with Groups of Prime Order. Proceedings of the 2nd International Conference on Pairing-Based Cryptography, Egham, UK, 2008, 75-88.
8. J. Bethencourt, A. Sahai, B. Waters. Ciphertext-Policy Attribute-Based Encryption. Proceedings of the 2007 IEEE Symposium on Security and Privacy, 2007, 321-334.
9. E. Shen, E. Shi, B. Waters. Predicate Privacy in Encryption Systems. Proceedings of the 6th Conference on Theory of Cryptography, San Francisco, 2009, 457-473.
10. GAS Lab Universita deli Studi di Salerno. jPBC Library. Accessible from (Last Accessed 5/18/2012) http://gas.dia.unisa.it/projects/jpbc/index.htm
11. S. Jarecki, V. Shmatikov. Efficient two party secure computation on committed inputs. EuroCrypt 2007. LNCS 4515, 97-114.
12. C. Gentry. Fully homomorphic encryption using ideal lattices. In Proceedings of the 41st annual ACM symposium on Theory of computing (STOC '09). New York, NY, 169-178.
13. J. Kilian. Founding Cryptography on Oblivious Transfer. Proceedings of the 20th Annual ACM Symposium on the Theory of Computation (STOC), 1988.
14. Apache Software Foundation. ActiveMQ. Accessible from (Last Accessed 5/18/2012) http://activemq.apache.org/features.html.
15. J. Bethencourt, A. Sahai, B. Waters. CP-ABE Library. Accessible from (Last Accessed 5/18/2012) http://acsc.cs.utexas.edu/cpabe.
16. P. Stuedi, I. Mohammed, M. Balakrishnan, Z. Morley Mao, V. Ramasubramanian, D. Terry, T. Wobber. Contrail: Enabling Decentralizing Social Networks on Samrtphones. Proceedings of Middleware 2011, Lisboa, LNCS 7049, 41-60.
17. R. Ostrovsky, W. E. Skeith. Private Searching on Streaming Data. Journal of Cryptology, 20, 4 (October 2007), 397-430.