Efficient and Private Three-Party Publish/Subscribe

Giovanni Di Crescenzo[1], Jim Burns[1], Brian Coan[1], John Schultz[3], Jonathan Stanton[3], Simon Tsang[1], and Rebecca N. Wright[2]

[1] Applied Communication Sciences, NJ, USA
{gdicrescenzo,bcoan,stsang,jburns}@appcomsci.com
[2] Rutgers University, NJ, USA
rebecca.wright@rutgers.edu
[3] Spread Concepts, MD, USA
{jschultz,jonathan}@spreadconcepts.com

Abstract. We consider the problem of modeling and designing publish/subscribe protocols that safeguard the privacy of clients' subscriptions and of servers' publications while guaranteeing efficient latency in challenging scenarios (i.e., real-time publication, high data arrival rate, etc.). As general solutions from the theory of secure function evaluation protocols would not achieve satisfactory performance in these scenarios, we enrich the model with a third party (e.g., a cloud server). Our main result is a three-party publish/subscribe protocol suitable for practical applications in such scenarios because the publication phase uses only symmetric cryptography operations (a result believed not possible without the third party). At the cost of only a very small amount of privacy loss to the third party, and with no privacy loss to the publishing server or the clients, our protocol has very small publication latency, which we measured for large parameter ranges to be just a small constant factor worse than a publish/subscribe protocol guaranteeing no privacy.

1 Introduction

Publish/subscribe protocols address the problem of publishing data items to interested participants. In a simple formulation of the problem, a publish/subscribe protocol can be considered a protocol between multiple clients, each with its own interests, and multiple servers with data items and associated topics. The servers would like to distribute a data item to a client if there is a match between the data item's topics and the client's interests. These protocols come in many different formulations and variations, as well surveyed in [1], and find applications in a large number of areas. In many applications, however, privacy is a sensitive issue that may deter from the implementation or use of a publish/subscribe system. For instance, in finance, a publish/subscribe system that allows clients to receive quotes from a stock market server, while revealing the clients' interests, may not only impact clients' privacy but also significantly alter the stock market pricing process and overall integrity.

In this paper, we investigate the modeling and design of publish/subscribe protocols with satisfactory levels of both privacy and efficiency in a challenging scenario of high arrival-rate data and real-time publishing. First, we note that designing a private

J. Lopez, X. Huang, and R. Sandhu (Eds.): NSS 2013, LNCS 7873, pp. 278–292, 2013.

publish/subscribe protocol in the two-party model (i.e., with no third party) using general solutions from the area of secure function evaluation protocols (e.g., [2]) would not meet our efficiency targets, one reason being that such protocols require public-key cryptographic primitives [3], which are significantly more expensive than their private-key cryptography counterparts. Departing from the two-party model and considering a three-party model helps towards efficiency. General solutions in this three-party model (i.e., client, server and third party), such as [4–6], would likely still be not efficient in our scenario because of significant resource requirements (e.g., interaction and/or randomness and/or cryptographic operations) for each gate and each input bit of the circuit associated with the publish/subscribe predicate. Instead, we consider the problem of designing efficient three-party publish/subscribe protocols, possibly at the expense of allowing some minimal privacy leakage to the third party (but not to the server or the clients and not about actual interests, topics or data items).

Our Contribution and Solution Sketch. Under this problem formulation, we design a publish/subscribe protocol that satisfies a highly desirable set of requirements: publication correctness (i.e. clients obtain a data item if their subscription predicate is satisfied by their interests and the data item's topics), privacy against malicious adversaries (i.e., a malicious adversary corrupting any one of client, server or third party cannot extract any information about interests, topics or data items) and efficiency (i.e., the publication, which is the real-time part of the protocol, only requires a small number of private-key cryptography operations).

Our protocol is natural and simple, and uses pseudo-random functions [7] and symmetric encryption as cryptographic primitives (but could be implemented using only information-theoretic tools). Our main technical contribution is that of representing client's interests and data item's topics using two-layer cryptographic pseudonyms, requiring only a few symmetric cryptography operations per (interest,topic) pair, and then directly performing computation over such pseudonyms, by testing equality statements without need for cryptographic operations. The computation of the topic pseudonyms is performed by the server during publication (with a randomizer specific to the data item) and the computation of the interest pseudonyms is split into two phases: the 1st layer is computed by the client during subscription (with a client-specific randomizer) and the 2nd layer is computed by the third party during publication (and given the appropriate randomizer by the server). During publication, the third party can evaluate the client's subscription predicate using interest and topic pseudonyms, *without any further cryptographic operation*. A high-level description can be found in Figure 1.

We prove privacy properties of our protocol using a natural adaptation of the real/ideal security definition approach (frequently used in cryptography), and show that our protocol leaks no information to server and clients, and only minimal information to the third party: the structure of each client's subscription predicate (but not the client's interests) and how many (interest,topic) pairs match. We also describe measurements of the protocol's publication latency, which, for large and practical parameter ranges, is only a small (≤ 6) constant slower than a publish/subscribe system with no privacy.

Related Work. Although there are a number of interesting publish/subscribe protocols with various security or privacy properties (e.g., [8–14]), they do not our combined functionality and privacy requirements for a mixture of reasons, including: a differ-

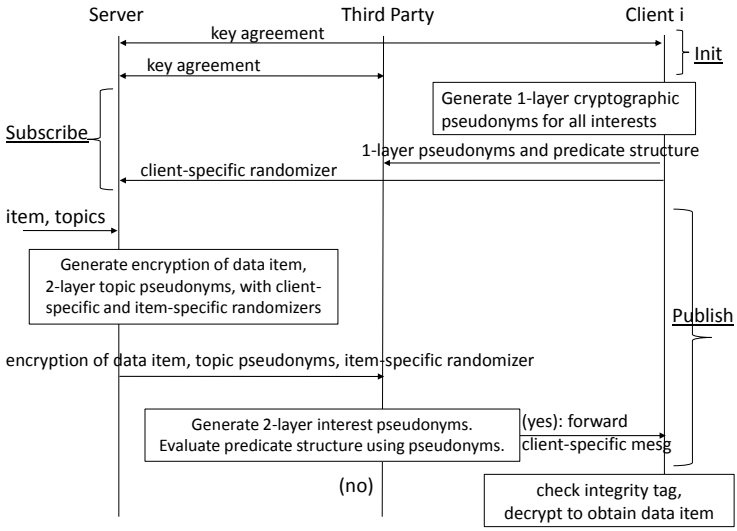

Fig. 1. Informal description of our publish/subscribe protocol

ent participant model (i.e., they typically consider entirely distributed models with no servers or third parties), and a different set of capabilities and functionalities (i.e., they typically target simple rules for content publication). Perhaps the closest solutions to our paper are [8, 12], which use essentially the same participant model as ours. To the best of our knowledge, no previous work presents rigorous modeling of security or privacy requirements for publish/subscribe systems or rigorous proofs that the proposed solutions meet any such requirements.

2 Models and Definitions

In this section we detail definitions of interest during our investigation of private publish/subscribe protocols: data, participant, topology, network and protocol models and publication correctness, privacy and efficiency requirements.

Data Model. We consider the following data objects or structures.

Data items. We represent the published data items as binary strings of length ℓ_d.

Dictionary and topics. To each data item, we associate d keyword tags, also denoted as *topics*, taken from a known set, called the *dictionary*, assumed, for simplicity, to be the set of all ℓ_t-bit strings. To each client, we associate c keyword tags, also denoted as *interests*, taken from the dictionary.

Subscription predicate: for $i = 1, \ldots, n$, a subscription from client C_i is formally represented as a boolean predicate, denoted as p_i, having equality statements of the type "$top_h = int_j$" as inputs, where top_h denotes the h-th topic associated with the current data item and int_j denotes the j-th interest associated with C_i, for $h \in \{1, \ldots, d\}$ and $j \in \{1, \ldots, c\}$. For each subscription predicate p_i, we define the associated *predicate*

structure ps_i as the representation of the predicate obtained by replacing each equality statement "$top_h = int_j$" with the pair (h, j). That is, each input to ps_i keeps pointers to the same topic and the same interest as in p_i, but does not explicitly contains the topic and interest strings. (This allows parties to have some workable representation of the predicate, without revealing the actual strings representing interests or topics).

Data items and associated topics are assumed to be streamed (at possibly large speed) to the server. Generalizations to other data arrival scenarios are possible, but not further discussed in this paper. We have, for simplicity, defined length and number variables ℓ_d, ℓ_t, d, c as system parameters with value known to all parties; however, smaller values for specific clients or data items can be accommodated by simple padding techniques.

Participant and Network Model. We consider the following types of participants, all assumed to be *efficient* (i.e., running in probabilistic polynomial-time in a common security parameter, denoted in unary as 1^σ). A *client* is a party that submits subscription updates based on his interests and a specific subscription predicate; we assume there are n parties, denoted as $C_1, \ldots, C_n$; a generic client may also be denoted as C. The *server* is the party, denoted as S, processing submitted data items (and associated topics) and client interests to realize the publish/subscribe functionality. The *third party*, denoted as TP, helps clients and servers to carry out their functions.

Each client is assumed to be capable of communicating with both the server and the third party. All clients are capable to be communicating with each other, but are not required to do so in our proposed protocol. For simplicity, we consider a confidential and authenticated network (this assumption is without loss of generality as parties can use a security protocol like TLS) with no packet loss. Additionally, we also restrict to the scenario where server and third party are assumed to be always connected to the network; clients are allowed to temporarily disconnect from the network (and thus potentially not receive matching data items while disconnected).

Protocol Model. A publish/subscribe protocol includes the following subprotocols:

Init: S and TP may exchange messages with $C_1, \ldots, C_n$, to initialize their data structures and/or cryptographic keys. Formally, on input a security parameter 1^σ, protocol Init returns private outputs for all parties, denoted as $out_S^{in}, out_{TP}^{in}, \{out_C^{in} : \forall C\}$.

Subscribe: C submits his updated subscription (based on C's set of interests and a subscription predicate) to S (and possibly TP) who update their record of C's subscription. Formally, on input a security parameter 1^σ to all parties, and a set of interests $int_1, \ldots, int_c$ and a subscription predicate p_i as private inputs of client C_i, for some $i \in \{1, \ldots, n\}$ protocol Subscribe returns private outputs for all participants, denoted as $out_S^{su}, out_{TP}^{su}, \{out_C^{su} : \forall C\}$.

Publish: S distributes the data item to each client based on the item's topics and the clients' interests and subscription predicate, possibly in collaboration with TP. In terms of distribution strategy, this protocol follows the so-called 'push mode': as soon as a new data item arrives, it is processed by S and TP and eventually sent to the appropriate subset (or all) of the clients. Formally, on input a security parameter 1^σ to all parties, and a data item m and a set of topics $top_1, \ldots, top_d$ as private inputs of server S, protocol Publish returns a (possibly empty) data item m as private output for (possibly a subset of the) clients and additional private outputs for all participants, denoted as

$out_S^{pu}, out_{TP}^{pu}, \{out_C^{pu} : \forall C\}$. Generalizations to other distribution strategies, like the so-called 'pull mode', are possible but not further discussed in this paper.

Requirements. Let σ be a security parameter. A function over the set of natural numbers is *negligible* if for all sufficiently large $\sigma \in \mathcal{N}$, it is smaller than $1/p(\sigma)$, for any polynomial p. Two distribution ensembles $\{D_\sigma^0 : \sigma \in \mathcal{N}\}$ and $\{D_\sigma^1 : \sigma \in \mathcal{N}\}$ are *computationally indistinguishable* if for any efficient algorithm A, the quantity $|\text{Prob}[\, x \leftarrow D_\sigma^0 : A(x) = 1\,] - \text{Prob}[\, x \leftarrow D_\sigma^1 : A(x) = 1\,]|$ is negligible in σ (i.e., no efficient algorithm can distinguish if a random sample came from one distribution or the other). A participant's *view* in a protocol (or a set of protocols) is the distribution of the sequence of messages, inputs and internal random coins seen by the participant while running the protocol (or the set of protocols). We address publish/subscribe protocols that satisfy the following classes of requirements: *correctness* (i.e., correctness of publication of data items to clients with matching predicate and interests), *privacy* (i.e., privacy of data items, interests and topics against all protocol participants, and of the subscription predicate against the third party), and *efficiency* (i.e., minimal time, communication and round complexity). We will use the following requirements.

Publication Correctness: for each data item m and associated topics $top_1, \ldots, top_d$, each client C_i with subscription predicate p_i and interests $int_1, \ldots, int_c$, the probability ϵ that, after an execution of Init on input 1^σ, an execution of Subscribe on input $int_1, \ldots, int_c, p_i$, and an execution of Publish on input $m, top_1, \ldots, top_d$, one of the following two events happens, is negligible in σ: (a) predicate p_i is satisfied by interests $int_1, \ldots, int_c$ and topics $top_1, \ldots, top_d$ but $out_{C_i}^{pu} \neq m$; (b) predicate p_i is not satisfied by interests $int_1, \ldots, int_c$ and topics $top_1, \ldots, top_d$ but $out_{C_i}^{pu} = m$.

Privacy: We use a natural adaptation of the real/ideal and universal composability (see, e.g., [15]) security frameworks, which are commonly used in the cryptography literature. Assume an environment E that delivers private inputs and randomness to all parties, as needed in the publish/subscribe protocol lifetime. For any efficient (i.e., probabilistic polynomial time) adversary Adv corrupting one of the three party types (i.e., client C, server S or third party TP), there exists an efficient algorithm Sim (called the *simulator*), such that for any efficient environment algorithm E, Adv's view in the "real world" and Sim's output in the "ideal world" are is computationally indistinguishable to E, where these two worlds are defined as follows. In the *real world*, runs of the Init, Subscribe and Publish subprotocols are executed, while Adv acts as the corrupted party. In the *ideal world*, each run of the Init, Subscribe and Publish subprotocols is replaced with an 'ideal execution' that is specifically designed to only reveal some 'minimal information', in addition to system parameters, inputs and outputs based on the publish/subscribe funtionality and related condition (see, e.g.,[16]). Here, we choose this minimal information to be the predicate structure ps_i and the evaluation results of the 'interest = topic' equality statements inputs to ps_i for TP (and no additional information for C and S). Thus, we define these ideal executions of Init, Subscribe and Publish as follows:

1. Ideal-Init, on input security parameter 1^σ, returns all system parameters and an *ok* string to all participants.
2. Ideal-Subscribe, on input a predicate p and a sequence of c interests $int_1, \ldots, int_c$ from C, returns a predicate structure ps to TP and an *ok* string to C, S and TP.

3. Ideal-Publish, on input a data item m and a sequence of d topics $top_1, \ldots, top_d$ of known length from S, returns an *ok* string to S and the following for each client C_i: the data item m to C_i if predicate p_i is satisfied by C_i's interests and m's topics $top_1, \ldots, top_d$; and the following to TP: the predicate structure ps_i and bits b_{hj} denoting which pairs (h, j) input to ps_i satisfy "topic(h)=interest(j)" (or not).

Efficiency: The protocol's *latency* is measured as the time taken by a sequential execution of subprotocols Init, Subscribe, Publish (as a function of σ and other system parameters). The protocol's *communication complexity* (resp., *round complexity*) is defined as the length (resp., number) of the messages, as a function of σ and other system parameters, exchanged by C, S and TP during subprotocols Init, Subscribe, Publish. Even if we will mainly focus our analysis on publication latency, our design targets minimization of all the mentioned efficiency metrics.

Although we have focused our formalization on the correctness, privacy and efficiency properties, we note that our design has targeted a number of additional *security* properties, which are however obtained using well-known techniques. Specifically, properties like *confidentiality* of the communication between all participants, message *sender authentication*, message *receiver authentication*, and *communication integrity* protection, can be immediately obtained by using a security protocol like TLS. Other simple and inexpensive steps to add security properties (i.e., to prevent TP to modify the encryption of the data item received by S before transferring it to the appropriate clients) are directly discussed in the presentation of our protocol. In the rest of this document, we describe our protocol, prove that it satisfies the above correctness and privacy requirements, and show some runtime analysis of its efficiency properties.

3 A Simple and Efficient Publish/Subscribe Protocol

In this section we describe our publish/subscribe protocol. We start with a formal statement of the properties of our protocol, then discuss the known and new cryptographic primitives used in the protocol, and give an informal description, a detailed description, and a proof of the properties of our protocol.

Theorem 1. In the model of Section 3.1, there exists (constructively) a publish/subscribe protocol satisfying the following properties:

1. publication correctness with error negligible in security parameter σ;
2. privacy against adversary Adv corrupting S, under no unproven assumption;
3. privacy against adversary Adv corrupting C, under no unproven assumption;
4. privacy against adversary Adv corrupting TP, assuming that F is a family of pseudo-random functions and (KG,E,D) is a secure symmetric encryption scheme.

An important claim of our paper is that our protocol, in addition to satisfying Theorem 1, has desirable performance on all efficiency metrics: round complexity, communication complexity, subscription latency, and, especially, publication latency. Our testing experiments and results on the latter metric can be found in Section 3.2.

3.1 Cryptographic Primitives and Properties Used

Our publish/subscribe protocols use the following cryptographic primitives or tools or approaches: pseudo-random functions [7], symmetric encryption schemes, and 2-layer cryptographic pseudonyms.

Pseudo-random Functions and Secure Symmetric Encryption Schemes. A pseudo-random function F [7] maps a key $k \in \{0,1\}^\kappa$ and an input x to an output $y \in \{0,1\}^\ell$, for some values κ, ℓ suitably related to the security parameter σ, and with the property that to any efficient algorithm making queries to an oracle O, the case $O = F(k, \cdot)$, when k is randomly chosen, is computationally indistinguishable from the case $O = R(\cdot)$, for a random function R with input and output of the same length. For our results, F could be realized using standard cryptographic tools like block ciphers or cryptographic hashing.

A symmetric encryption scheme [17] is a triple (KG,E,D), where KG, the key generation algorithm, returns a key k on input a security parameter 1^κ; E, the encryption algorithm, returns a ciphertext c on input a key k and a message m; D, the decryption algorithm, returns a plaintext m' on input a key k and a ciphertext c. For our results, (KG,E,D) can be realized using textbook schemes based on block ciphers and pseudo-random functions, which satisfy well accepted security notions such as security in the sense of indistinguishability against chosen ciphertext attacks.

Two-Layer Cryptographic Pseudonyms. To protect the privacy of clients' interests and data item's topics, we use cryptographic pseudonyms (possibly involving repeated applications of F) so to later allow TP to perform computation directly on cryptographic pseudonyms, instead of the individual interest and topic bits (as done in other techniques like secure function evaluation). To enable equality checks between client interests and item topics by the third party, the interests and topics' pseudonyms will be defined using the same pseudonym function pF, consisting of repeated application of F, and defined as follows: on input x, function pF returns

$$F(k_{s,tp}, F(k_{s,c(i)}, x|r_i)|s),$$

where $k_{s,c(i)}$ is a key shared between S and C_i, $k_{s,tp}$ is a key shared between S and TP, r_i is a client specific randomizing nonce, and s is a data item specific randomizing nonce. Building on [18], cryptographic pseudonyms use keys shared by different parties and achieve the following: C can generate 1-layer interest pseudonyms, S can generate topic pseudonyms, TP can check whether an interest pseudonym is equal to a topic pseudonym, and leakage of both interests and topics to TP is prevented. Furthermore, the computation of key $k_{s,c(i)}$ is re-randomized at each execution of the Subscribe protocol and for each interest, using a random counter ctr and computing $k_{s,c(i),j} = F(k_{s,c(i)}, ctr+j)$, for $j = 1, \ldots, c$. We note that the function pF satisfies the following

Lemma 1. If F is a pseudo-random function the following holds: (1) if interest int_j and topic top_h are equal, then so are the associated interest pseudonym $pF(int_j)$ and topic pseudonym $pF(top_h)$; (2) if the interest int_j and topic top_h are distinct, then the associated interest pseudonym $pF(int_j)$ and topic pseudonym $pF(top_h)$ are computationally indistinguishable from two random and independent strings of the same length. (Hence, they are not different only with negligible probability).

Proof of Lemma. Part (1) of this fact follows from the fact that $pF(int_j)$ is computed from int_j in the same way as $pF(top_h)$ is computed from top_h (i.e., using a triple application of F, based on the same counter ctr, and the same randomizing nonces r_i, s, and the same keys $k_{tp,s}, k_{s,c(i),j}$). Part (2) of this fact follows by observing that when $int_j \neq top_h$, the function pF is pseudo-random (as so is F) and, when evaluated on two distinct inputs, returns two outputs that are computationally indistinguishable from two random strings of the same length. □

In our publish/subscribe protocol, TP can compute 2-layer interest pseudonyms, with help from client and server, and receive 2-layer topic pseudonyms from the server. Later, it can then evaluate the client's subscription predicate using interest and topic pseudonyms as input, without further cryptographic operations. By Lemma 1, this is equivalent, except with negligible probability, to evaluating the client's predicate p_i on input interests and topics, but without any leakage of information about interests or topics to any unintended parties. Depending on the result of the predicate evaluation, TP sends or does not send an encrypted version of the data item to the client, who decrypts it.

The privacy of interests and topics is guaranteed by the computation of cryptographic pseudonyms via pseudo-random functions. The privacy of the data item is guaranteed by use of encryption. We avoid TP to learn correlations among interests in the same subscription (e.g., if the same interest is used more than once) by using an independent key $k_{s,c(i),j}$, computed using a key $k_{s,c(i)}$ and a random counter ctr_i, to compute the pseudonym for each $j = 1, \ldots, c$. We avoid TP to learn correlations among interests in different subscriptions (e.g., if the same interest is used on two different subscriptions) by randomizing the pseudonym computation with random nonce r_i. We avoid TP to learn correlations among topics in different data items (e.g., if the same topic appears on two different data items) by randomizing the pseudonym computation with random nonce s. We achieve high efficiency on publication latency as the Publish subprotocol only requires highly efficient symmetric-key computations.

3.2 Detailed Description

We proceed with a formal description of our publish/subscribe protocol (see Figure 2 for a pictorial description, however omitting some steps for better clarity).

Preliminaries: This protocol assumes a point-to-point secure communication protocol such as TLS to be used for all exchanged communication, and suitable message headers including protocol name, subprotocol name, and unique session, sender and receiver ID's. While for simplicity of presentation, we always refer to a single client C in the description below, we note that in our multiple-client scenario, each client runs C's program described below (using independently chosen random strings), and the other parties repeat their program, described below, for each of the clients (again, using independently chosen random strings).

Init: Server S sets a key length parameter κ (e.g., $\kappa = 128$). Then S and each client C_i, for $i = 1, \ldots, n$, run a secure key-agreement protocol to jointly generate a random and independent key $k_{s,c(i)} \in \{0,1\}^\kappa$ (such a protocol can be built using standard cryptographic protocols [19] or even just requiring S to choose a key and send it to C_i).

Analogously, S and third party TP run a secure key-agreement protocol to jointly generate a random and independent key $k_{s,tp} \in \{0,1\}^\kappa$. As a result of these subprotocols, one symmetric key is shared by S and C_i but not by TP, and one key is shared by S and TP but not by any of $C_1, \ldots, C_n$. Moreover, these keys will actually be used as inputs to a pseudo-random function to generate, using standard techniques (e.g., a counter and a block cipher like AES), an arbitrarily large number of pseudo-random keys with the same property (i.e., being shared by only two of the parties).

Subscribe: Let C be a client with interests $int_1, \ldots, int_c$, and a subscription predicate p with predicate structure ps. In this operation, S, TP and client C_i, for some $i \in \{1, \ldots, n\}$, run the following instructions:

1. C_i uniformly and independently chooses a random nonce $r_i \in \{0,1\}^\ell$ and a random starting counter $ctr_i \in \{0,1\}^\ell$
2. For $j = 1, \ldots, c$, C_i computes pseudo-random key $k_{s,c(i),j} = F(k_{s,c(i)}, ctr_i + j)$ and 1-layer interest pseudonym $ip^i_{j,1} = F(k_{s,c(i),j}, (int_j | r_i))$
3. C_i sends the current subscription predicate structure ps_i and 1-layer pseudonyms $(ip^i_{1,1}, \ldots, ip^i_{c,1})$ to TP
4. C_i sends (r_i, ctr_i) to S
5. TP replaces C_i's 1-layer interest pseudonyms with the just received $(ip^i_{1,1}, \ldots, ip^i_{c,1})$
6. TP replaces C_i's subscription predicate structure with the just received ps_i
7. S replaces C_i's random nonce and counter with the just received (r_i, ctr_i)

Publish: We assume that S receives a new data item m, with topics $top_1, \ldots, top_d$. In this operation, involving S, TP and clients $C_1, \ldots, C_n$, the parties run the following instructions:

1. S uniformly and independently chooses a nonce $s \in \{0,1\}^\ell$
2. S computes data item ciphertext $M = E(k, m)$ and $K_i = E(k_{s,c(i)}, k)$, for $i = 1, \ldots, n$
3. For $j = 1, \ldots, c$ and $i = 1, \ldots, n$,
   S computes $k_{s,c(i),j} = F(k_{s,c(i)}, ctr_i + j)$, using last ctr_i received from C_i
4. For each $h = 1, \ldots, d$, $j = 1, \ldots, c$ and $i = 1, \ldots, n$,
   S computes 1-layer topic pseudonym $tp^i_{h,j,1}$ as $= F(k_{s,c(i),j}, top_h | r_i))$
   S computes 2-layer topic pseudonym $tp^i_{h,j,2}$ as $= F(k_{s,tp}, (tp^i_{h,j,1} | s))$
5. S computes tags $tag_i = F(k_{s,c(i)}, M | K_i)$, for $i = 1, \ldots, n$
6. S sends $(M, s, \{tp^i_{h,j,2} : h, j, i\}, \{(K_i, tag_i) : i = 1, \ldots, n\})$ to TP
7. For $i = 1, \ldots, n$,
   for $j = 1, \ldots, c$,
   TP computes 2-layer interest pseudonyms $ip^i_{j,2} = F(k_{s,tp}, (ip^i_{j,1} | s))$
   TP evaluates ps_i on input $\{tp^i_{h,j,2} : h, j, i\}$ and $\{ip^i_{j,2} : j = 1, \ldots, c\}$
   if ps_i evaluates to 1, then TP sends (M, K_i, tag_i) to C_i
   if C_i receives a message (M, K_i, tag_i),
   if $tag_i \neq F(k_{s,c(i)}, M | K_i)$ then C_i returns: "error" and halts.
   else C_i computes $k = D(k_{s,c(i)}, K_i)$, $m = D(k, M)$ and returns: m.

In the rest of this section we discuss why our protocol satisfies publication correctness, privacy and efficiency properties, as defined in Section 2.

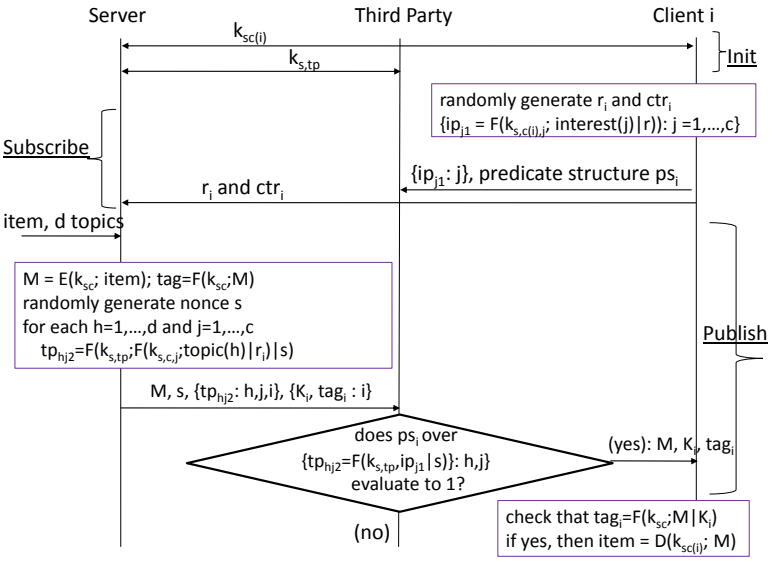

Fig. 2. Our publish/subscribe Protocol

3.3 Properties: Correctness, Privacy and Efficiency

Publication Correctness. We observe that a client C_i receives data item m from TP if the subscription predicate structure ps_i returns 1 when its input equality statements are evaluated over the interest and topic pseudonyms (rather than the interests and topics themselves). However, we note that ps_i returns the same value, except with negligible probability, regardless of whether the equality statements are evaluated over the interest/topic pseudonyms or over the interests/topics. This latter claim, implying the publication correctness property, is implied by observing that there is a polynomial number of interests and topics and by an application of Lemma 1.

Privacy. We achieve privacy against a malicious adversary Adv that corrupts any one of the participants; i.e., either S, or TP, or a client C_i. The protocol only leaks values of global parameter (i.e., length parameters) and the intended protocol functionality outputs (i.e., the data items to the matching clients) to clients or server. To the third party, the protocol only leaks the following: the client's predicate structure, but not the interests, (here, we note that it is not unreasonable for a practical system to have the client's predicate structure as a known protocol parameter), and the bits b_{hj} denoting whether the j-th interest in a client's subscription is equal to the h-th topic associated with a data item (without revealing anything else about interests, topics or data items). Actually, our proof extends to malicious adversaries that corrupts all clients or a subset of them. We divide the formal proof into 3 cases, depending on whether Adv is corrupting S, TP, or a client C_i. In all cases, the simulation of the Init protocol directly follows from the simulation properties of the key agreement protocol used. Thus, we only

focus on the Subscribe and Publish subprotocols. Let $m(S, TP)$ denote the message $(M, s, \{tp^i_{h,j,2} : h, j, i\}, \{(K_i, tag_i) : i = 1, \ldots, n\})$ sent from S to TP.

Case Adv=S: Assume an adversary, denoted as Adv, corrupts S. For any such Adv, we show a simulator Sim that produces a view for Adv in the ideal world (while posing as S) that is computationally indistinguishable from Adv's view in the real world (while posing as S), during the execution of the Init, Subscribe and Publish protocols.

To simulate Adv's view in the Subscribe subprotocol, Sim invokes the ideal Subscribe functionality, which only returns an ok string to S, and then randomly chooses a randomizing nonce r_i and a random starting counter ctr_i for each client C_i, for $i = 1, \ldots, n$. Then Sim simulates the subscription message from C_i to S as (r_i, ctr_i).

To simulate Adv's view in the Publish subprotocol, on input the data item m and the associated topics $top_1, \ldots, top_d$, Sim invokes the ideal Publish functionality, which only returns an ok string to S, and then runs Adv on input $m, top_1, \ldots, top_d$ to obtain a message $m(S, TP)$. If S does not return such a message, then Sim simply halts.

We note that for all three protocols, the simulation from Sim is perfect, in that the distribution of Sim's output (representing Adv's view in the ideal world) and the distribution of Adv's view in the real world are the same.

Case Adv=TP: Assume an efficient malicious adversary, denoted as Adv, corrupts TP. For any such Adv, we show a simulator Sim that produces a view for Adv in the ideal world (while posing as TP) that is computationally indistinguishable from Adv's view in the real world (while posing as TP), during the execution of the Init, Subscribe and Publish protocols.

To simulate Adv's view in the Subscribe subprotocol, Sim invokes the ideal Subscribe functionality, which returns client C_i's subscription predicate structure ps_i to TP, and sends ps_i to Adv. Moreover, Sim randomly and independently chooses values $ip^{i,\cdot}_{j,1} \in \{0,1\}^\ell$, for $j = 1, \ldots, c$, and sends them to Adv.

Finally, to simulate Adv's view in the Publish subprotocol, Sim invokes the ideal Publish functionality, which returns to TP bits b^i_{hj} denoting whether equality "topic(h) = interest(j)" in predicate p_i is satisfied or not. Then, to simulate message $m(S, TP)$, Sim simulates each value in this message's tuple either as a suitable encryption of a random value of the appropriate length (which is known as it is a protocol parameter) or as the output of the appropriate length (also known as a protocol parameter) of a pseudo-random evaluation, as follows. First of all, Sim randomly chooses keys $k', s', k'_{s,c(1)}, \ldots, k'_{s,c(n)}$, a data item m', and hash tags $tag'_1, \ldots, tag'_n$, and computes an encryption $M' = E(k', m')$ and encryptions $K'_i = E(k'_{s,c(i)}, k')$, for $i = 1, \ldots, n$. Furthermore, to simulate the topic pseudonyms, Sim considers each equation "topic(h)=interest(j)" in predicate structure ps_i, for each $i = 1, \ldots, n$. If $b^i_{hj} = 0$ (i.e., the equation does not hold), then Sim uniformly chooses a value $tp^{i,\cdot}_{h,j,2}$. If $b^i_{hj} = 1$ (i.e., the equation holds), then Sim sets value $tp^{i,\cdot}_{h,j,2} = ip^{i,\cdot}_{j,2} = F(k_{s,tp}, (ip^{i,\cdot}_{h,j,1}|s))$. Finally, Sim can simulates $m(S, TP)$ as $(M', s', \{tp^{i,\cdot}_{h,j,2} : h, j, i\}, \{(K'_i, tag'_i) : i = 1, \ldots, n\})$ and the message by TP to clients by simply running Adv's program.

We now show that Sim's output (i.e., Adv's view in the ideal world) and Adv's view in the real world are computationally indistinguishable. With respect to the simulation of subprotocols Subscribe and Publish, we can prove that the simulation is

computationally indistinguishable from Adv's view in the real world, as follows. First, we observe that the only differences between the two views are the following:

1. the value M, an encryption of data item m, in Adv's view vs. the value M', an encryption of a random value of the same length, in Sim's output: this difference is proved to be computationally indistinguishable by using the security property of the used symmetric encryption scheme;
2. the values $tag_1, \ldots, tag_n$, where tag_i is a MAC tag of (M, K_i), for $i = 1, \ldots, n$, in Adv's view vs. the randomly chosen values $tag'_1, \ldots, tag'_n$ in Sim's output: this difference is proved to be computationally indistinguishable by using the pseudo-randomness property of the used function F;
3. the 1-layer interest pseudonyms $ip^i_{j,1}$ in Adv's view vs. the randomly chosen values $ip^{i,\cdot}_{j,1}$ in Sim's output: this difference is proved to be computationally indistinguishable by using the pseudo-randomness property of the used function F;
4. conditioned on the interest pseudonyms, the topic pseudonyms $tp^i_{h,j,2}$ in Adv's view vs. the values $tp^{i,\cdot}_{h,j,2}$ in Sim's output: these values are equally distributed when $b^i_{hj} = 1$ (i.e., the equation holds) since they are computed in the same way in both spaces, and are proved to be computationally indistinguishable when $b^i_{hj} = 1$ (i.e., the equation does not hold) by using the pseudo-randomness property of F.

We then observe that by combining the above observations and a standard hybrid argument [17], we can prove that the entire Sim's output and the entire Adv's view in the real world are computationally indistinguishable, assuming the pseudo-randomness property of F and the security of the symmetric encryption scheme used.

Case Adv=C: Assume an adversary, denoted as Adv, corrupts a client C_i. For any such Adv, we show a simulator Sim that produces a view for Adv in the ideal world (while posing as C_i) that is computationally indistinguishable from Adv's view in the real world (while posing as C_i), during the execution of the Init, Subscribe and Publish protocols. To simulate the Subscribe subprotocol, given as input interests $int_1, \ldots, int_c$ and predicate p_i, Sim invokes the ideal Subscribe functionality, which only returns an ok string to C_i, and invokes C to obtain the messages for TP and S. Finally, to simulate the Publish subprotocol, Sim invokes the ideal Publish functionality, possibly obtaining (or not) data item m and topics $top_1, \ldots, top_d$ as output for C_i, depending on whether the predicate p_i is satisfied by topics $top_1, \ldots, top_d$ and interests $int_1, \ldots, int_c$ or not. In the former case, Sim has to simulate the message M, K_i, tag_i from TP and can use data item m to do that perfectly, as follows. Sim computes $M' = E(k_{s,c(i),m})$, randomly chooses key $k' \in \{0,1\}^\kappa$, and computes $K'_i = E(k_{s,c(i)}, k')$ and $tag'_i = F(k_{s,c(i)}, (M'|K'_i))$. By inspection, we see that the simulation of subprotocol Init is perfect, in that the distribution of Sim's output and the distribution of Adv's view in the real world are the same.

Efficiency. While it is easy to verify that our protocol is very efficient on the communication complexity, round complexity and subscription latency metrics, it is of special interest to evaluate the publication latency metric, under varying parameter values. We implemented both our protocol, called P1, and a publish/subscribe protocol, called P0, that performs no additional cryptographic operation, other than using the TLS protocol

on all messages between parties. The testing was done on a collection of 6 Dell PowerEdge 1950 processors and one Dell PowerEdge 2950 processor. We divided clients in groups of size 25 each, and each group was run on each of 4 PowerEdge 1950 processors. The server was run on a dedicated 1950 processor, the third party was run on dedicated 1950 processor, and the testing control was run on the 2950 processor. All initialization, subscription, and publication traffic was run over a dedicated gigabit Ethernet LAN. Testing control and collection of timing measurement traffic was isolated on a separate dedicated gigabit Ethernet LAN.

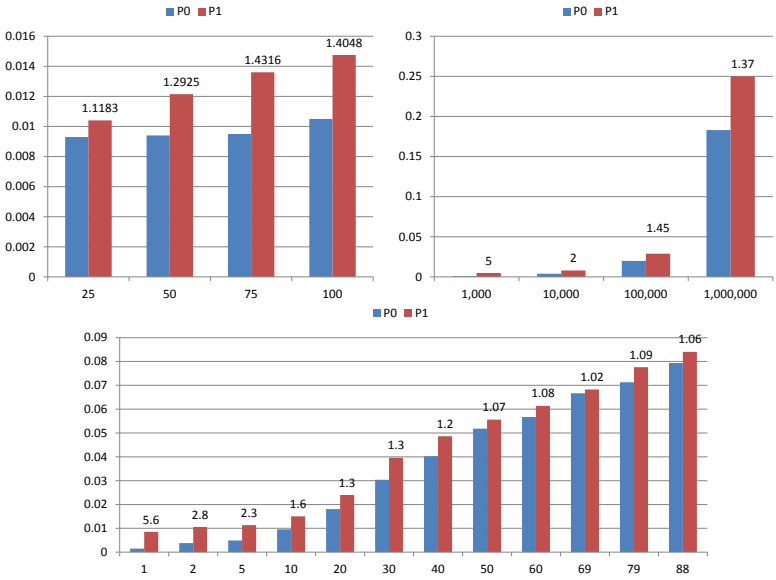

Fig. 3. Publication Latency Measurements for P1 and P0

We compared P1 and P0 under varying values for one of the following parameters: the total number of clients, the length of the data item and the number of matching clients. The initial parameter setting was: 100 clients, 10 matching clients per publication, 10 interests, 10 topics, and 1 publication of a 10K data item per second, where the matching predicate is the OR of all possible equalities between an interest and a topic. (We restricted to this predicate as in our protocol more complex predicates require no additional cryptographic operation other than TLS processing.) Under this setting, in Figure 3, the top left chart reports the max latency vs the number of clients when the latter varies from 25 to 100; the top right chart reports the max latency vs the size of the data item varying from 1K to 1M; the bottom chart reports the max latency vs the number of matching clients varying from 1 to 88. The labels on P1 columns indicate the ratio of the P1 latency to the P0 latency. In all three cases, the P1 latency is at most a small (1.5, 5, and 6, respectively) constant worse than the latency in P0 and scales well as the parameter increases.

Acknowledgements. This work was supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center (DoI/NBC) contract number D12PC00520. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation hereon. Disclaimer: 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.

References

1. Eugster, P.T., Felber, P., Guerraoui, R., Kermarrec, A.M.: The many faces of publish/subscribe. ACM Comput. Surv. 35(2), 114–131 (2003)
2. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: FOCS, pp. 162–167 (1986)
3. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: STOC, pp. 44–61 (1989)
4. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: STOC, pp. 218–229 (1987)
5. Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In: STOC, pp. 73–85 (1989)
6. Feige, U., Kilian, J., Naor, M.: A minimal model for secure computation (extended abstract). In: STOC, pp. 554–563 (1994)
7. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM 33(4), 792–807 (1986)
8. Raiciu, C., Rosenblum, D.S.: Enabling confidentiality in content-based publish/subscribe infrastructures. In: SecureComm, pp. 1–11 (2006)
9. Minami, K., Lee, A.J., Winslett, M., Borisov, N.: Secure aggregation in a publish-subscribe system. In: WPES, pp. 95–104 (2008)
10. Shikfa, A., Önen, M., Molva, R.: Privacy-preserving content-based publish/subscribe networks. In: Gritzalis, D., Lopez, J. (eds.) SEC 2009. IFIP AICT, vol. 297, pp. 270–282. Springer, Heidelberg (2009)
11. Tariq, M.A., Koldehofe, B., Altaweel, A., Rothermel, K.: Providing basic security mechanisms in broker-less publish/subscribe systems. In: DEBS, pp. 38–49 (2010)
12. Choi, S., Ghinita, G., Bertino, E.: A privacy-enhancing content-based publish/subscribe system using scalar product preserving transformations. In: Bringas, P.G., Hameurlain, A., Quirchmayr, G. (eds.) DEXA 2010, Part I. LNCS, vol. 6261, pp. 368–384. Springer, Heidelberg (2010)
13. Ion, M., Russello, G., Crispo, B.: Supporting publication and subscription confidentiality in pub/sub networks. In: Jajodia, S., Zhou, J. (eds.) SecureComm 2010. LNICST, vol. 50, pp. 272–289. Springer, Heidelberg (2010)
14. Pal, P., Lauer, G., Khoury, J., Hoff, N., Loyall, J.: P3S: A privacy preserving publish-subscribe middleware. In: Narasimhan, P., Triantafillou, P. (eds.) Middleware 2012. LNCS, vol. 7662, pp. 476–495. Springer, Heidelberg (2012)
15. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS, pp. 136–145 (2001)
16. Di Crescenzo, G., Ostrovsky, R., Rajagopalan, S.: Conditional oblivious transfer and timed-release encryption. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 74–89. Springer, Heidelberg (1999)

17. Goldwasser, S., Micali, S.: Probabilistic encryption. J. Comput. Syst. Sci. 28(2), 270–299 (1984)
18. Brickell, E., Di Crescenzo, G., Frankel, Y.: Sharing block ciphers. In: Clark, A., Boyd, C., Dawson, E.P. (eds.) ACISP 2000. LNCS, vol. 1841, pp. 457–470. Springer, Heidelberg (2000)
19. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (1976)