Privacy-Preserving Publish/Subscribe: Efficient Protocols in a Distributed Model

Giovanni Di Crescenzo[1(✉)], Brian Coan[1], John Schultz[2], Simon Tsang[1], and Rebecca N. Wright[3]

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

Abstract. We consider the problem of modeling and designing efficient and privacy-preserving publish/subscribe protocols in a distributed model where parties can act as publishers or subscribers or both, and there are no brokers or other types of parties. The problem is particularly challenging as privacy demands on such protocols come with efficiency limitations; most notably, the publisher must send messages as long as the publications to all parties, and the cryptographic techniques to perform the publish/subscribe match need to be based on asymmetric cryptographic operation which are known to be less efficient than their symmetric counterpart.

Our main result is a distributed publish/subscribe protocol which addresses and essentially nullifies the impact of both efficiency limitations, without sacrificing the required privacy properties. Our construction is based on very efficient design of a novel cryptographic tool, of independent interest, called 'hybrid conditional oblivious transfer protocol', as it resembles hybrid encryption, where asymmetric encryption is only used to transfer a short key, which enables (much more efficient) symmetric encryption of a long message.

1 Introduction

Publish/subscribe protocols address the problem of publishing data items to interested participants. They come in many different formulations and variations, as well surveyed in [1]. In this paper's formulation of the problem, a publish/subscribe protocol can be considered a distributed protocol between multiple participants who can, at any given time, act as subscribers (with subscription keywords, called interests) or publishers (with data items and related publication keywords, called topics). The publisher would like to distribute a data item to the subscribers if there is a match between the data item's topics and a subscriber's interests, with no help from brokers or other types of parties. These protocols find applications in a large number of areas, and are of

J. Garcia-Alfaro et al. (Eds.): DPM 2013 and SETOP 2013, LNCS 8247, pp. 114–132, 2014.
DOI: 10.1007/978-3-642-54568-9_8,

interest in essentially every area where distributed systems are used. In most applications, however, privacy is a sensitive issue that may even deter from the implementation or use of a publish/subscribe system. For instance, in finance, a publish/subscribe system assisting a market maker could allow subscribers to submit their interest in companies and publishers to issue data relative to companies; however, by revealing company names and data from either the subscribers or the publishers, it may not only impact participants' privacy but also significantly alter the market's pricing process and overall integrity.

In this paper we investigate the modeling and design of distributed publish/subscribe protocols which preserve the privacy of subscribers' interests and of publishers' data items and topics. We start by observing that such privacy demands come with at least two main efficiency limitations. First, while in non-private publish/subscribe protocols, a publisher can send data items only to matching subscribers, this cannot happen in protocols with privacy demands, as this would reveal the subset of matching subscribers (and thus, information about their interests) to the publisher. In fact, we make the rather discouraging observation that the publisher must send a message at least as long as the data item to each of the subscribers, regardless of whether the publication matches their interests or not. Second, computing which subscribers are entitled to data items can be shown, using well-known fundamental results in cryptography [2], to require asymmetric cryptographic operations, which are well-known to be less efficient than their symmetric counterparts, the difference being significant in applications with high data arrival rates, which are not uncommon publish/subscribe scenarios. In particular, general solutions from the area of secure function evaluation protocols (e.g., [3,4]) suffer from similar inefficiency drawbacks.

Our Contribution. We design a publish/subscribe protocol that not only addresses the mentioned efficiency limitations, but achieves desirable privacy and efficiency properties. Specifically, our protocol satisfies a highly desirable set of requirements: publication correctness (i.e. subscribers obtain a data item if their subscription predicate is satisfied by their interests and the data item's topics), privacy of interests (i.e., against a malicious adversary corrupting the publisher), privacy of topics and data items (i.e., against an honest-but-curious adversary corrupting even all the subscribers), and efficiency (i.e., the publication, which is the real-time part of the protocol, only requires a small rate of public-key cryptography operations per item). We overcome the two efficiency limitations as follows: first, we perform cryptographic processing of data items only once for all subscribers, by encrypting the data item once and distributing the key only to matching subscribers; then, we minimize the use of asymmetric cryptographic operations in distributing the encrypting key by using a novel hybrid cryptographic primitive (i.e., starting with asymmetric cryptographic operations and then continuing with symmetric ones for the rest of the protocol lifetime). Specifically, our protocol uses new constructions for conditional oblivious transfer (COT) protocols [5], called *hybrid COT protocols*, where the first execution of such a subprotocol requires asymmetric cryptography operations, while all

remaining ones, when based on the same private inputs, do not. We prove privacy properties using a natural adaptation of the real/ideal security definition approach (frequently used in cryptography), and show that our protocol leaks no information to the publisher or to all subscribers. We also describe measurements of the protocol's publication latency, which, for large and practical parameter ranges, is only a small (≤ 8) constant slower than a distributed publish/subscribe system with no privacy. Our techniques for hybrid COT protocols can also be extended to more general conditions than equality.

Related Work. Some papers have proposed interesting publish/subscribe protocols with some security or privacy properties (e.g., [6–11]). All these papers fall short of meeting our combined functionality and privacy requirements for a mixture of reasons, including a different set of security and/or privacy requirements (i.e., they often require privacy against intermediate routing nodes or privacy only against one party, or rely on trusted broker parties). None of these papers proves privacy properties in a formal, cryptographic model for private publish/subscribe protocols. Our previous paper in the area [12] proposes a solution with privacy provable in a cryptographic model but in a different participant model (i.e., using an intermediate broker to achieve even greater efficiency). We could not find any paper studying hybrid COT protocols; the seemingly closest paper [13] first studied a related problem about precomputing 1-out-of-2 oblivious transfer protocols, which is however different in at least 2 important ways (i.e., it is about 1-out-of-2 oblivious transfer instead of equality-based COT, and it performs several oblivious transfers in the preprocessing phase instead of one). Equality-based COT protocols were already presented in [5,14–17], which however did not consider the problem of designing hybrid constructions. The COT concept is a variant of oblivious transfer, which was first introduced by [18].

2 Models and Definitions

We detail models and definitions of interest during our investigation of private and distributed publish/subscribe protocols: data, participant, network and protocol models and correctness, privacy and efficiency requirements.

Data Model. We consider the following data objects or structures. The *data items* to be published are digital documents and are represented as binary strings of length ℓ_d. To each data item, we associate d publication keywords, also denoted as *topics*, taken from a set, called the *dictionary*, known to all parties, and assumed, for simplicity, to be the set of all ℓ_t-bit strings. To each party, we associate c subscription keywords, also denoted as *interests*, taken from the dictionary. Moreover, each party has a *public file* to post information accessible by all other participants (as for a public-key infrastructure in cryptography). Finally, each party has a list of all other system participants. For simplicity, length and number variables ℓ_d, ℓ_t, d, c are defined as system parameters with value known to all parties; however, smaller values can be accommodated by simple padding techniques. Data items and associated topics are assumed to be

either generated by or streamed to a publisher, at possibly high rate. Although we target high data arrival rates, we only deal with scenarios where an execution of the publish/subscribe protocol ends before the next data item is streamed to a publisher. Generalizations to other data arrival scenarios are possible, but not further discussed in this paper.

Participant and Network Model. We consider a distributed model with $n+1$ participants $P_1, \ldots, P_{n+1}$, all assumed to be *efficient* (i.e., running in probabilistic polynomial-time in a common security parameter, denoted in unary as 1^σ). Two participant roles are possible at any given time; specifically, a participant can act as a *publisher*, also denoted as P, when it publishes a data item to all other participants; or can act as a *subscriber*, also denoted as S_i, for $i \in \{1, \ldots, n\}$, when it posts its interests or receives another party's publication. Each participant is able to communicate with all others and post on its own public file (also implicitly defining a communication channel with all other parties) . 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 loss of transferred data or of party connectivity.

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

Init: participants $P_1, \ldots, P_{n+1}$, may interact and/or post messages on their public files to initialize their data structures and/or cryptographic keys. Formally, on input security parameter 1^σ, protocol Init returns public and secret outputs for all parties.

Subscribe: Party P_i, for $i \in \{1, \ldots, n+1\}$, acting as a subscriber, posts its updated subscription (based on its latest set of interests) on its public file, without interacting with any other party. Formally, on input security parameter 1^σ, a party index $i \in \{1, \ldots, n+1\}$, and a set of interests $int_1, \ldots, int_c$, algorithm Subscribe returns a public and a secret output, where the public output is posted on P_i's public file.

Publish: Party P_i, for $i \in \{1, \ldots, n+1\}$, acting as a publisher, distributes the data item to the subscribers (i.e., all remaining n participants) based on the data item's topics and on the other participants' subscriptions. In terms of distribution strategy, we consider a protocol that follows the so-called 'push mode': as soon as a new data item arrives, along with its topics, it is processed by the publisher towards the subscribers. Formally, on input 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 publisher P, protocol Publish returns a private output for the i-th subscriber, for $i \in \{1, \ldots, n+1\}$, which is either empty or equal to the data item m. Generalizations to other distribution strategies, like the so-called 'pull mode', are possible but not further discussed in this paper.

Requirements. We now briefly describe publication correctness, privacy and efficiency 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. We say that a subscriber S_i is *entitled* to data item m if at least one of subscriber S_i's interests $int_1, \ldots, int_c$ is equal to any one of the topics $top_1, \ldots, top_d$ associated with m. We address publish/subscribe protocols that satisfy the following classes of requirements.

Correctness. The probability of the following two events is negligible in the security parameter: (a) after executing Init and Subscribe, S_i is entitled to m but does not receive m as output from Publish; (b) after executing Init and Subscribe, S_i is not entitled to m but S_i receives m as output from Publish. Formally, for each data item m and associated topics $top_1, \ldots, top_d$, each subscriber S_i with 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 $int_1, \ldots, int_c$, 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) at least one of subscriber S_i's interests $int_1, \ldots, int_c$ is equal to at least one of the topics $top_1, \ldots, top_d$ but S_i's output at the end of the publication subprotocol is $\neq m$; (b) at least one of subscriber S_i's interests $int_1, \ldots, int_c$ is equal to at least one of the topics $top_1, \ldots, top_d$ but S_i's output at the end of the publication subprotocol is $= m$.

Privacy: We consider two privacy requirements: against a potentially malicious publisher, and against a coalition of honest-but-curious subscribers (i.e., subscribers who follow the protocol but can perform arbitrary computation at the end in their attempt to violate privacy properties). First, consider an efficient and potentially malicious publisher; we require that after an execution of protocols Init, Subscribe and Publish, any such participant learns no additional information about the subscribers' interests. Second, consider a coalition of efficient and honest-but-curious subscribers who did not subscribe to a data item m; we require that after an execution of protocols Init, Subscribe and Publish, any such coalition learns no additional information about the data item or its associated topics.

Towards a formal definition, we recall the notions of computational indistinguishability and participant's view. Two distribution ensembles $\{D^0_\sigma : \sigma \in \mathcal{N}\}$ and $\{D^1_\sigma : \sigma \in \mathcal{N}\}$ are *computationally indistinguishable* if for any efficient algorithm A, the quantity $|\mathrm{Prob}[\, x \leftarrow D^0_\sigma : A(x) = 1\,] - \mathrm{Prob}[\, x \leftarrow D^1_\sigma : A(x) = 1\,]|$ 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 use a natural adaptation of the real/ideal privacy definition framework, which is commonly used in the cryptography literature. A formal definition for the privacy requirement according to this framework goes, briefly speaking, as follows. For any efficient (i.e., probabilistic polynomial time) adversary *Adv* corrupting one of the two party types (i.e., either a publisher P or some subset of all subscribers $S_1, \ldots, S_n$), there exists an efficient algorithm *Sim* (called the *simulator*), such that *Adv*'s view in the "real world" and *Sim*'s output in the "ideal world" are computationally indistinguishable, where these two worlds

are defined as follows. In the *real world*, runs of the Init subprotocol, Subscribe algorithm and Publish subprotocol are executed, while Adv acts as the corrupted participant(s). In the *ideal world*, each run of the Init subprotocol, Subscribe algorithm and Publish subprotocol is replaced with an 'ideal execution' that does not reveal any additional information, in addition to system parameters, inputs and outputs intended by the publish/subscribe functionality. 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 a *done* string to all participants.
2. Ideal-Subscribe, on input a sequence of c interests $int_1, \ldots, int_c$ from a subscriber S_i, returns a *done* string to S_i.
3. Ideal-Publish, on input a data item m and a sequence of d topics $top_1, \ldots, top_d$ of known length from a publisher P, returns the data item m to each subscriber S_i for which at least one of S_i's interests is equal to at least one of the topics $top_1, \ldots, top_d$, and a *done* string to all remaining subscribers and publisher P.

Efficiency: The protocol's *latency* is measured as the time taken by a sequential execution of subprotocol Init, algorithm Subscribe, and subprotocol 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 publisher and subscribers during subprotocols Init, Publish. Even if we will mainly focus our efficiency analysis on publication latency, our design targets minimization of all the mentioned efficiency metrics.

We observe that in any protocol satisfying privacy against the publisher, the latter cannot tell if a subscriber receives the data item or not. Because this holds regardless of the distribution of the data item's content, it also holds for random data items, which cannot be compressed. We thus obtain the following

Proposition 1. In any publish/subscribe protocol in our model, satisfying privacy against the publisher, in the Publish protocol, the publisher needs to send at least ℓ_d bits to each subscriber.

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.

3 Hybrid Conditional Oblivious Transfer

In this section we formally define the notion of hybrid COT protocols, and then design one such protocol for the equality condition, under the intractability of the Decisional Diffie-Hellman problem.

Equality Conditional Oblivious Transfer (eq-COT): Definition. Informally, an eq-COT protocol is a 2-party protocol where a sender wants to privately transfer a message to a receiver in a way that the only leaked information is the sender's message when the equality predicate evaluates to 1 with private inputs from sender and receiver. Here, we slightly adapt the formal definition from [5] to consider the equality predicate and to more easily express the hybrid COT definition later. Then an eq-COT protocol is a pair (Alice,Bob) of probabilistic polynomial algorithms where Alice's (respectively, Bob's) private input is a string x_a (resp., x_b); m_a denotes Alice's message, m_b denotes Bob's output at the end of the protocol, and the following requirements hold: *(Transfer Correctness)* if $x_a = x_b$ then the probability that $m_b \neq m_a$ is negligible; if $x_a \neq x_b$ and m_a is uniformly distributed, then the distribution of m_b is uniform and independent from m_a; *(Privacy against Bob)* if $x_a \neq x_b$ then for any efficient adversary *Adv* corrupting Bob, the protocol's communication transcript reveals no information to *Adv* about m_a; *(Privacy against Alice)* for any efficient adversary *Adv* corrupting Alice, the protocol's communication transcript reveals no information to *Adv* about whether $x_a = x_b$ or not.

Hybrid Equality Conditional Oblivious Transfer (h-eq-COT). Informally, an h-eq-COT protocol is a 2-party, 2-phase, protocol that allows Alice to perform an eq-COT of an arbitrary number of messages to Bob, as follows. In a first phase, called *h-eq-COT protocol, asymmetric phase*, Alice and Bob execute a single preliminary eq-COT of a κ-bit random symmetric key k_a, based on asymmetric cryptography techniques, where k_a denotes Alice's input and k_b denotes the key received by Bob at the end of this phase. In a second phase, called *h-eq-COT protocol, symmetric phase*, Alice and Bob execute an eq-COT of a message m_a, based on symmetric cryptography techniques, where Alice takes as input k_a, m_a and Bob takes as input k_b and receives m_b at the end of this phase. That is, in all symmetric phase executions of the eq-COT protocol Alice and Bob take as input the symmetric key returned at the end of the preliminary eq-COT protocol (i.e., the same key if $x_a = x_b$ or random and independent keys otherwise.) The formal definition of an h-eq-COT protocol is derived by extending the one for an eq-COT protocol and is omitted here.

Our h-eq-COT Protocol. Similarly to almost all known efficient 1-out-of-2 oblivious transfer (OT) protocols (e.g., [16,17,19]), we base our hybrid COT protocol on an encryption scheme with suitable malleability and/or homomorphism properties. In particular, we use the Decisional Diffie-Hellman problem [20] and its properties, as done in El-Gamal encryption [21] and in the 1-out-of-2 OT protocol from [19], the latter is well known to have especially the latter having desirable security and performance properties.

Informally speaking, the h-eq-COT protocol can be described as follows. First, in the preliminary eq-COT protocol, Bob posts an asymmetric encryption of string x_b, where the encryption scheme used allows Alice to later manipulate this encryption and transform it, without knowing x_b, into an encryption of $k_b = k_a(x_b/x_a)^r \bmod p$, for some random value r, where k_a is Alice's input secret key. In this way, if $x_b = x_a$, Bob receives an encryption of $k_b = k_a$, which he

can decrypt; while if $x_b \neq x_a$, Bob receives an encryption of a random key k_b independently distributed from k_a. More formally, this preliminary eq-COT protocol goes as follows:

1. Using a public random source, Alice and Bob uniformly and independently choose σ-bit primes p, q such that $p-1$ is a multiple of q, a generator g for the q-order subgroup G_q of Z_p, and a random key $k_h \in \{0,1\}^\kappa$ that defines an efficiently invertible map $M_{\kappa,p}$ from $\{0,1\}^\kappa$ to G_q
2. Bob computes $x'_b = M_{\kappa,p}(k_h, x_b)$, where $x'_b \in G_q$
3. Bob randomly chooses $r_0, r_1 \in Z_q$, computes $h = g^{r_0} \bmod p$, $u = g^{r_1} \bmod p$ and $v = h^{r_1}(x'_b) \bmod p$ and sends (h, u, v) to Alice
4. Alice computes $x'_a = M_{\kappa,p}(k_h, x_a)$, where $x'_a \in G_q$; randomly chooses $k_a \in \{0,1\}^\kappa$ and computes $k'_a = M_{\kappa,p}(k_h, k_a)$, where $k'_a \in G_q$; and randomly chooses $s_0, s_1 \in Z_q$
5. Alice computes $w = g^{s_0} u^{s_1} \bmod p$ and $z = h^{s_0}(v/x'_a)^{s_1} \cdot k'_a \bmod p$ and sends w, z to Bob
6. Bob computes $k'_b = zw^{-r_0} \bmod p$, and $k_b = M^{-1}_{\kappa,p}(k_h, k'_b)$, for $k_b \in \{0,1\}^\kappa$
7. Bob returns: k_b.

At any later time, to perform an eq-COT transfer of any message m, Alice uses key k_a and Bob uses key k_b, and both use an arbitrary symmetric encryption scheme, denoted as (KG,E,D), where E (resp., D) is the encryption (resp., decryption) algorithm. Alice can just perform a symmetric encryption of data item m based on k_a and Bob would be able to decrypt the right item whenever $k_b = k_a$, which holds whenever Alice's private input x_a is equal to Bob's private input x_b. For efficiency purposes, we use another session key so that Bob does not need to decrypt the (potentially long) message when the decryption is not successful. More formally, this preliminary eq-COT protocol goes as follows:

1. Alice randomly chooses a session key $skey_a \in \{0,1\}^\kappa$
2. Alice computes an encryption of message m as $M = E(skey_a, m_a)$, and values $c = E(k_a, skey_a)$ and $tag = E(skey_a, 0^\kappa)$, and sends (M, c, tag) to Bob
3. Bob computes $skey_b = D(k_b, c)$ and checks if $tag = E(skey_b, 0^\kappa)$;
   if not, Bob returns: $\perp$.
   if yes, Bob computes $m_b = D(skey_b, M)$ and returns: m_b.

We also designed variants of the above constructions based on [16,17], but we omit them here, as they seemed slightly less efficient.

Properties. Building on results from [5,19,21], we obtain the following properties for the above h-eq-COT protocol:

1. If Alice and Bob are honest, and $x_a = x_b$, then at the end of the protocol the value m_b obtained by Bob is equal to the value m_a transferred by Alice.
2. If Alice is honest, and $x_a \neq x_b$, then for any polynomial-time adversary *Adv* corrupting Bob, at the end of the protocol, *Adv* learns no additional information about Alice's input x_a or the message m_a.

3. The message (h, u, v) from Bob to Alice can be efficiently simulated by returning a random triple from $(G_q)^3$, and the simulated triple is computationally indistinguishable from the same triple in the real execution assuming the intractability of the Decisional Diffie-Hellman problem. This implies that any polynomial-time adversary corrupting Alice does not learn anything about x_b.
4. When $x_a = x_b$, the messages (w, z) and (M, c, tag) sent by Alice to Bob can be efficiently simulated against an adversary corrupting Bob and having x_b as input and obtaining m_b as output, and the simulation's output is distributed exactly as in the real execution;
5. When $x_a \neq x_b$, the messages (w, z) and (M, c, tag) sent by Alice to Bob can be efficiently simulated against an adversary corrupting Bob and having x_b as input, and the simulation's output is computationally indistinguishable from the same messages in the real execution, assuming the security of the used encryption scheme (KG,E,D).

Proof of properties. We now sketch a proof of properties 1-5 of our h-eq-COT protocol.

Proof of property 1. To see that property 1 is satisfied, we prove two facts: (a) if Alice and Bob are honest and $x_a = x_b$ then at the end of h-eq-COT, asymmetric phase, it holds that $k_b = k_a$; (b) if Alice and Bob are honest and $k_a = k_b$ then at the end of h-eq-COT, symmetric phase, it holds that $m_b = m_a$.

To prove (a), observe that $x_a = x_b$ implies $x'_a = x'_b$ and thus

$$z = h^{s_0}(v/x'_a)^{s_1} \cdot k'_a = h^{s_0}(h^{r_1}x'_b/x'_a)^{s_1} \cdot k'_a = h^{s_0+r_1 s_1} \cdot k'_a = g^{r_0 s_0 + r_0 r_1 s_1} \cdot k'_a \bmod p.$$

Then we have that

$$w^{-r_0} = (g^{s_0} u^{s_1})^{-r_0} \bmod p = g^{-r_0 s_0 - r_0 r_1 s_1} \bmod p,$$

from which we see that $k'_b = z w^{-r_0} \bmod p = k'_a$, which implies that

$$k_b = M^{-1}_{\kappa,p}(k_h, k'_b) = M^{-1}_{\kappa,p}(k_h, k'_a) = k_a.$$

To prove (b), observe that $k_a = k_b$ implies that

$$skey_b = D(k_b, c) = D(k_a, c) = D(k_a, E(k_a, skey_a)) = skey_a$$

and therefore $E(skey_b, 0^\kappa) = E(skey_a, 0^\kappa)$ and

$$m_b = D(skey_b, M) = D(skey_a, M) = D(skey_a, E(skey_a, m_a)) = m_a.$$

Proof of Property 2 (Sketch). Similarly as for property 1, we can show that when $x_a \neq x_b$, for any h, u, v sent by an adversary playing as Bob, we have that $k'_b = k'_a \cdot (x'_b/x'_a)^{s_1} \bmod p$, for some $x'_b = v h^{-r_1} \bmod p$. Since s_i is random, we have that k'_b is random and independent from k'_a, and thus cannot be used by Adv to obtain any information about m_a or x_a from the message (M, c, tag) sent by Alice.

Proof of Property 3 (Sketch). This property follows by the observation that the (u, v) is an El-Gamal encryption of x'_b and thus the well-known fact that the tuple (g, h, u, v) is computationally indistinguishable from a random tuple from $(G_q)^4$.

Proof of Property 4 (Sketch). To prove this property, we now show a simulator *Sim* that, when $x_a = x_b$, efficiently simulates the messages (w, z) and (M, c, tag) sent by Alice to Bob, and using x_b and m_b as input, and show that the simulation's output is distributed exactly as in the real execution.

Sim generates (w, z) exactly as Alice does, with the only apparent difference that it uses x_b instead of x_a. *Sim* can do that since it has the exact same inputs k_a and x_a as Alice, and we are considering the case $x_a = x_b$. Specifically, *Sim* randomly chooses $s_0, s_1 \in Z_q$, and $k_a \in \{0,1\}^\kappa$, and generates w, z as $w = g^{s_0} u^{s_1} \bmod p$ and $z = h^{s_0}(v/x'_b)^{s_1} \cdot k'_a \bmod p$, where $x'_b = M_{\kappa,p}(k_h, x_b)$ and $k'_a = M_{\kappa,p}(k_h, k_a)$.

Analogously, *Sim* generates (M, c, tag) exactly as Alice does, with the only apparent difference that it uses m_b instead of m_a. *Sim* can do that since it has the exact same inputs k_a and m_a as Alice, and we are considering the case $x_a = x_b$, which implies that $m_a = m_b$.

Proof of Property 5 (Sketch). To prove this property, we now show a simulator *Sim* that, when $x_a \neq x_b$, efficiently simulates the messages (w, z) and (M, c, tag) sent by Alice to Bob, using x_b as input, and show that the simulation's output is computationally indistinguishable from the same messages in the real execution, assuming the security of the used encryption scheme (KG,E,D).

Sim generates w, z as two random and independent values in Z_p. By using an analogue property of the oblivious transfer protocol from [19], we obtain that the output of this simulation is equally distributed to the same pair in the real execution.

Moreover, *Sim* generates (M, c, tag) as encryptions of random messages of the same length of the messages encrypted in the real execution. By a standard hybrid argument, this triple is computationally indistinguishable from the triple generated in the real execution, assuming the security of the used encryption scheme (KG,E,D).

4 A Distributed Publish/Subscribe Protocol

In this section we describe our distributed 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 Sect. 2, there exists (constructively) a distributed publish/subscribe protocol satisfying the following properties: (1) publication correctness with error negligible in security parameter σ; (2) privacy against any efficient adversary corrupting a publisher P, under the hardness of the Decisional

Diffie-Hellman problem; (3) privacy against any efficient and honest-but-curious adversary corrupting an arbitrary subset of subscribers, under the security of the symmetric encryption scheme (KG,E,D) used; (4) non-interactive subscription; (5) one-message publication.

An important claim of our paper is that our protocol, in addition to satisfying Theorem 1, has highly desirable publication latency. In our testing experiments we verified that for a large domain of practical parameter values, the publication latency of our protocol remains within a small constant factor (i.e., 8) worse than the publication latency of a protocol performing the same functionality but offering no privacy guarantee. An example chart for these results is described at the end of this section.

4.1 Informal Description

Our goal is to design a distributed publish/subscribe protocol where the subscription phase is non-interactive (i.e., each subscriber simply posts a message on its public file), the publication protocol requires a single message from publisher to subscribers, and where the publisher is allowed to be malicious and the subscribers are allowed to collude in their attempt to violate the privacy requirements, as specified in Sect. 2.

A high-level view of our protocol can be given as follows. During the initialization subprotocol, the parties agree on common cryptographic parameters using publicly available randomness. During the subscription phase, a subscriber simply runs an asymmetric encryption algorithm to compute an encryption of each one to its interests, and posts such encryptions on its public file. During the publication phase, a publisher sends a single message to all subscribers so that this message, combined with the instructions run by a subscriber, form a conditional oblivious transfer of the data item to be published. Here, the condition is the subscription predicate (i.e., at least one of the subscriber's interests is equal to at least one of the data item's topics). This is reduced to running, for each (data item topic, subscriber interest) pair, an equality-COT where the condition is equality between the data item topic and the subscriber's interest in this pair.

Now, a main goal in the design of our protocol is to minimize the use of asymmetric cryptographic primitives, which are well known to be less efficient than their symmetric counterpart. Specifically, to minimize this efficiency degradation, we use them in a way that is reminiscent of the very practical 'hybrid encryption' approach, where an asymmetric encryption scheme is only used once per communication session to establish the key for a symmetric encryption scheme, and the latter is used for all message encryptions required in the future. Then, we realize a 'hybrid' equality-COT protocol where for each publisher and subscriber, the first of such transfers for a given (data item topic, subscriber interest) pair is performed using asymmetric primitives and all following ones for the same pair re-use the symmetric key established during the first one, using memoization. Then we use an hybrid equality-COT, as described in Sect. 3, which uses: (1) for

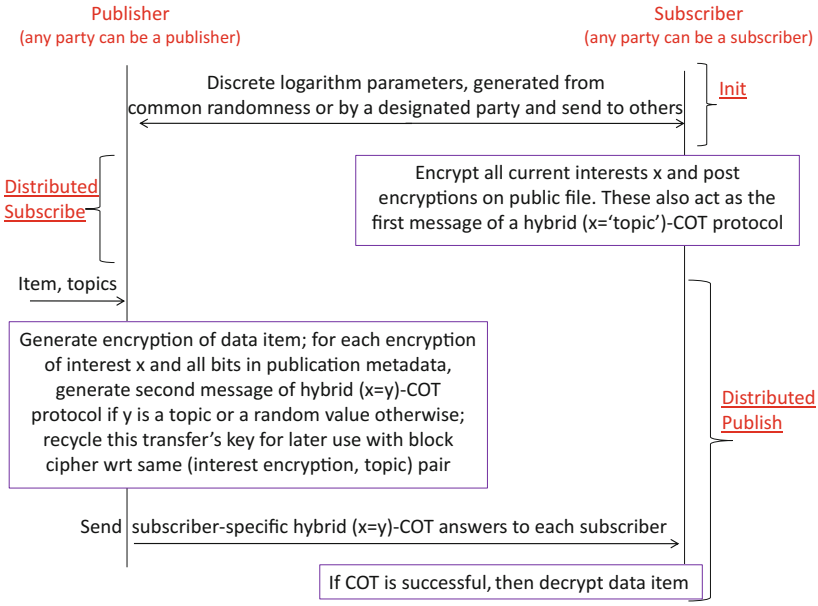

Fig. 1. Informal description of our publish/subscribe protocol

the symmetric part, an equality-COT based on symmetric encryption; and (2) for the asymmetric part, El-Gamal encryption [21] and a novel variant of the most efficient known oblivious transfer protocol [19]. Using asymmetric encryption helps, among other things, avoiding low-entropy guessing attacks on the subscribers' interests and publisher's topics.

An informal pictorial description of our protocol can be found in Fig. 1.

4.2 Detailed Description

We proceed with a formal description of our distributed publish/subscribe protocol (see Fig. 2 for a pictorial description).

Protocol Preliminaries: A point-to-point secure communication protocol such as TLS is assumed to be used for all exchanged communication.

Init: In the initialization subprotocol, parties $P_1, \ldots, P_{n+1}$ run the following instructions:

1. Let ρ be a sufficiently long random string available to all parties; if such a string is not available, $P_1, \ldots, P_{n+1}$ run a multi-party key-agreement protocol to generate one
2. $P_1, \ldots, P_{n+1}$ use ρ to generate the triple (p, q, g) as defined in the initialization subprotocol of the h-eq-COT protocol

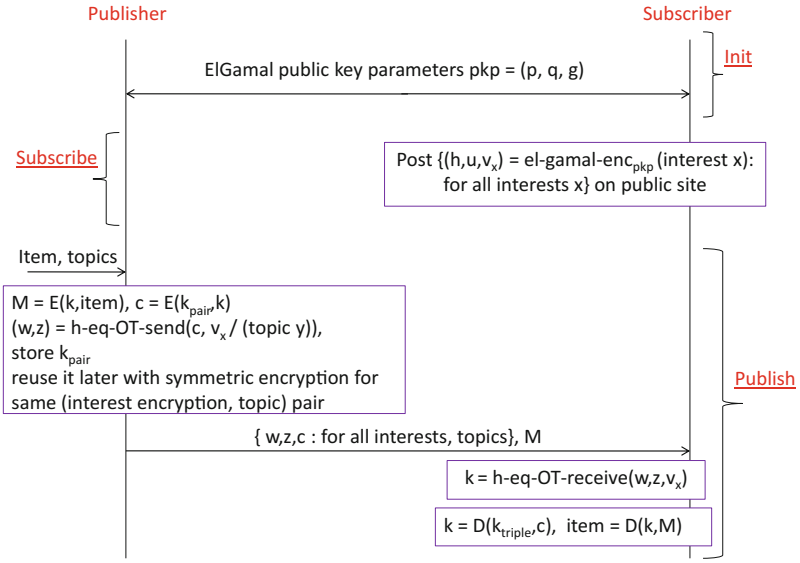

Fig. 2. Our publish/subscribe protocol

Subscribe: Recall that a subscriber S_i's subscription is formally represented as a sequence of c interests $int_1, \ldots, int_c$, for some integer $c \geq 1$. To subscribe, S_i runs the following instructions:

1. For $j = 1, \ldots, c$,
   let int_j denote subscriber S_i's jth interest
   S_i uses triple (p, q, g) to compute a value h_j and an asymmetric encryption (u_j, v_j) of int_j, as done in h-eq-COT protocol, asymmetric phase, step 1
   S_i sets $ip_j = (h_j, u_j, v_j)$
2. S_i posts $(ip_1, \ldots, ip_c)$ on its public file

Publish: We assume that a participant, acting as a publisher P, somehow originates a new data item m, associated with a number d of topics. In the Publish subprotocol, involving P and all remaining participants, acting as subscribers $S_1, \ldots, S_n$, the following instructions are repeated for each subscriber $S_i, i = 1, \ldots, n$.

1. P computes a random key $k_p \in \{0,1\}^\kappa$, and an encryption of data item m as $M = E(k_p, m)$, and sends M to S_i
2. P computes $tag = E(k_p, 0^\kappa)$, and sends tag to S_i
3. For $h = 1, \ldots, d$,
   for $j = 1, \ldots, c$,
   each current interest pseudonym ip_j from C_i, where $j = 1, \ldots, c_p$,
   if P and S_i had not yet executed the key transfer for this
   (interest encryption (u_j, v_j), topic top_h) pair,

P randomly chooses $k_{pair,p} \in \{0,1\}^\kappa$
P uses the h-eq-COT protocol, asymm. phase, to transfer $k_{pair,p}$ to S_i
let $k_{pair,s,i}$ be the key received by S_i at the end of this subprotocol
P uses the h-eq-COT protocol, symm. phase, to transfer k_p to S_i,
where P uses $k_{pair,p}$ and S_i uses $k_{pair,s,i}$ as additional input
let $k_{s,i}^{h,j}$ be the key received by S_i at the end of this subprotocol
if P and S_i had already executed the key transfer for this
(interest encryption (u_j, v_j), topic top_h) pair,
P uses the h-eq-COT protocol, symm. phase, to transfer k_p to S_i,
where P uses $k_{pair,p}$ and S_i uses $k_{pair,s,i}$ as additional input
let $k_{s,i}^{h,j}$ be the key received by S_i at the end of this subprotocol

4. S_i checks if $tag = E(k_{s,i}^{h,j}, 0^\kappa)$ for some $h \in \{1, \ldots, d\}$ and $j \in \{1, \ldots, c\}$
5. if yes, then S_i computes $m = D(k_{s,i}^{h,j}, M)$ for the found h, j values, and returns: m; else S_i returns: $\perp$

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

4.3 Properties: Correctness, Privacy and Efficiency

Publication Correctness: To prove that our protocol satisfies this requirement, we need to show the facts (a) and (b) as from the requirement definition.

To see that fact (a) is satisfied, assume that one of subscriber S_i's interests, denoted as int_h, is equal to one of the data item's topics, denoted as top_j. Then, by Property 1 of the h-eq-COT protocol, when run on input top_j as Alice's input and int_h as Bob's input, the key $k_{s,i}^{h,j}$ received by S_i, when playing as Bob, is equal to the key k_p sent by P, when playing as Alice, and used to encrypt the data item m as M. Accordingly, S_i can successfully decrypt M and receive the data item m with probability 1.

To see that fact (b) is satisfied, assume that all of subscriber S_i's interests are different from all of the data item's topics. Then, by Property 2 of the h-eq-COT protocol, when run on input a topic as Alice's input and an interest as Bob's input, all keys $k_{s,i}^{h,j}$ received by S_i, when playing as Bob, are random and independent from the key k_p sent by P, when playing as Alice, and used to encrypt the data item m as M. Accordingly, S_i can successfully decrypt M and receive the data item m with probability smaller than $(cd)\delta(\sigma)$, for some negligible function δ, which is negligible in σ.

Privacy. Our protocol achieves privacy against an efficient and potentially malicious adversary that corrupts the publisher and against an efficient and honest-but-curious adversary that corrupts any subset of the subscribers. Accordingly, we divide the proof of this property into these two cases. In both cases, the simulation of the Init protocol directly follows from the simulatability properties of the key agreement protocol used (if necessary). Thus, we only focus on the simulation of the output of the Subscribe algorithm and of the Publish subprotocol.

Here, in both cases, which we now discuss, we show the existence of an efficient simulator algorithm that simulates Adv's view.

Adv corrupts P: In this case, the simulation mainly follows from the simulation specified in Property 3 of our h-eq-COT protocol. Specifically, assume an efficient adversary, denoted as Adv, corrupts the publisher P. For any such Adv, we show a simulator Sim that produces a view for Adv in the ideal world (while posing as P) that is computationally indistinguishable from Adv's view in the real world (while posing as P), during the execution of the Init, Subscribe and Publish protocols.

To simulate Adv's view from the subscription phase, Sim invokes the ideal Subscribe functionality, which only returns a *done* string to P. Then, the messages posted by the subscribers on their public file are generated by Sim as in the simulation specified in Property 3 of our h-eq-COT protocol.

To simulate Adv's view in the Publish subprotocol, on input the data item m and topics $top_1, \ldots, top_d$, Sim invokes the ideal Publish functionality, which returns a *done* string to P, and then runs Adv on input $m, top_1, \ldots, top_d$ to obtain P's messages to all subscribers. If P does not return such a message, then Sim simply halts.

The proof that Sim's simulation in the ideal world is computationally indistinguishable from Adv's view in the real world, follows from Property 3 of our h-eq-COT protocol, which holds under the intractability of the Decisional Diffie-Hellman problem.

Adv Corrupts an Arbitrary Subset of Subscribers: In this case, the simulation mainly follows from the simulation specified in Properties 4 and 5 of our h-eq-COT protocol. Specifically, assume an efficient and honest-but-curious adversary, denoted as Adv, corrupts a subset of subscribers, or even all of them. For any such Adv, we show a simulator Sim that produces a view for Adv in the ideal world (while posing as the corrupted subscribers) that is computationally indistinguishable from Adv's view in the real world. To simulate the output of the Subscribe algorithm, given as input interests $int_1, \ldots, int_c$ for each corrupted subscriber, Sim does the following. It invokes the ideal Subscribe functionality, which only returns a *done* string to all subscribers. Then it invokes the corrupted subscribers to directly obtain the message they post on their public file. Finally, it simulates the messages posted by the uncorrupted subscribers on their public file, exactly as done in the previous case; that is, again using the simulation specified in Property 3 of our h-eq-COT protocol.

Finally, to simulate the Publish subprotocol, Sim invokes the ideal Publish functionality, possibly obtaining (or not) data item m as output for the corrupted subscribers, depending on whether at least one of the topics $top_1, \ldots, top_d$ is equal to at least one of the interests $int_1, \ldots, int_c$ or not, for each specific subscriber in the corrupted subset. In the former case, Sim has to simulate the strings (w, z) and (M, c, tag) sent by P and can use data item m to do that perfectly, by running P's algorithm. Specifically, Sim runs the simulator as specified in Property 4 of our h-eq-COT protocol. In both cases, Sim can simulate the strings sent to the subscribers by P as part of each execution of the h-eq-COT

protocol by running the simulator as specified in Property 4 of our h-eq-COT protocol, where it is also proved that the simulation from Sim is equally distributed to Adv's view in the real world.

In the latter case, Sim has to again simulate the strings (w, z) and (M, c, tag) sent by P but does not have data item m this time. However, Sim can run the simulator as specified in Property 5 of our h-eq-COT protocol. Here, the proof that the simulation from Sim in the ideal world is computationally indistinguishable from Adv's view in the real world, follows from Property 5 of our h-eq-COT protocol, which holds under the security of the encryption scheme used.

Efficiency. By inspection, we verify that in our publish/subscribe protocol the subscription is non-interactive, in that each subscriber only posts a message on its public file (which is from then on readable by any publisher), and the publication only requires a single message from a publisher to all subscribers. Our protocol's efficient communication complexity is also easy to verify.

It remains of interest to evaluate the publication latency metric, under varying parameter values. We implemented both our protocol, denoted as P3.0, and another publish/subscribe protocol, called P0, that only addresses communication privacy and integrity properties using the TLS protocol on all messages between parties, and does not address any privacy on interests, topics or data items between publisher and subscribers.

We note that in P0 the publisher only communicates to the matching subscribers, while this cannot happen in P3.0 or otherwise its privacy property would be violated, due to Proposition 1. Thus, it is of interest to ask whether we can avoid the communication overhead of sending an encryption of the data item to all subscribers, especially in applications where the data item is large. Accordingly, we also implemented a protocol, denoted as P3.1, as the following variant of P3.0:

1. During the Publish subprotocol, the publisher sends the encryption $E(k, m)$ of the data item to a third party, called *repository server*, together with a random value $t = E(k, nonce)$, acting as an access token.
2. The publisher runs the same, previously defined, Publish subprotocol, this time publishing token t instead of a data item
3. All subscribers that received key k and token t send t to the repository server
4. If the repository server receives a valid access token from a subscriber, he sends to this subscriber the associated encryption $E(k, m)$ of the data item
5. The subscriber uses k to retrieve the data item.

We performed testing on a collection of 6 Dell PowerEdge 1950 processors and one Dell PowerEdge 2950 processor. Subscribers were divided in 4 groups of size 25 each, and each group was run on a PowerEdge 1950 processor. The publisher 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.

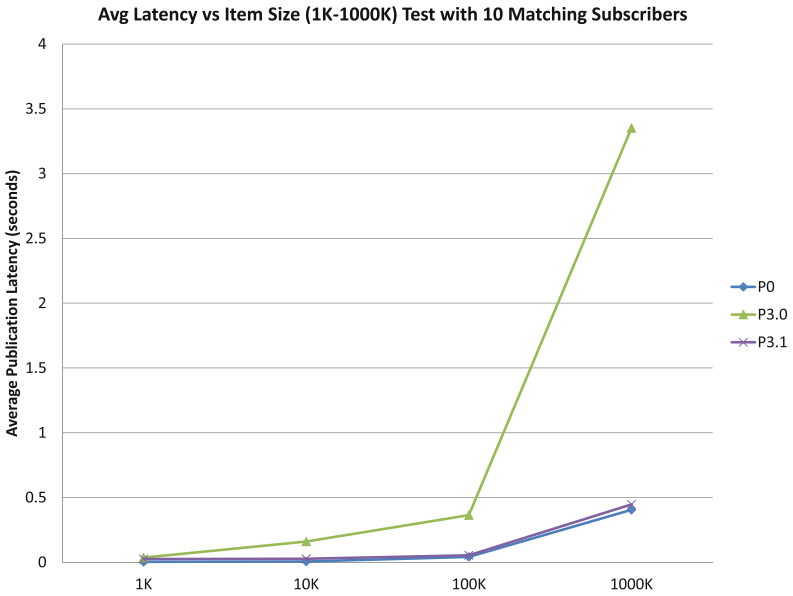

Fig. 3. Publication latency measurements for P3.0, P3.1, and P0

We compared P3.0. P3.1 and P0 against three sets of parameters, each test with a publication rate of 1 item per second, with values 1 K, 10 K, 100 K, and 1000 K bytes for data items, with 10 matching subscribers and 10 topics per item. (see Fig. 3). Tests were run dozens of times over a period of several weeks and results were consistently the same over all the runs (i.e. there is nothing stochastic in the experiments). A resulting performance chart can be found in Fig. 3.

In particular, we observe that the performance of the P3.0 protocol is always within a small constant (e.g., ≤ 8) of the performance of P0. This is remarkable, in light of the fact that P3.0 sends information to all participants (to safeguard privacy against the publisher), while P0 is only sending the data item to the interested participants. Our protocol P3.1, employing a repository server, has performance very close to that of P0 (in other words, it achieves high performance and moderately satisfactory privacy properties, but at the cost of having to trust the repository server).

5 Conclusions

We formally defined a distributed model for publish/subscribe protocols where participants can act as publishers or as subscribers in any given publication transaction. In this challenging model, we showed that solutions with provable privacy and efficiency are possible. In particular, two inherent efficiency limitations (the use of asymmetric cryptography operations and the fact that data

items need to be sent to all subscribers) can be mitigated to have only a very small impact on performance, allowing private solutions with efficiency comparable to non-private solutions. This is achieved without the need of a broker (as required in our recent solution [12]). Our approach, based on a novel cryptographic primitive (i.e., hybrid conditional oblivious transfer protocols), can also be generalized to more elaborate publish/subscribe conditions.

Acknowledgements. Many thanks go to Jim Burns and Jonathan Stanton for useful technical conversations. 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, PTh, Felber, P., Guerraoui, R., Kermarrec, A.-M.: The many faces of publish/subscribe. ACM Comput. Surv. **35**(2), 114–131 (2003)
2. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations.In: Proceedings of the ACM STOC, pp. 44–61 (1989)
3. Yao, A.C.-C.: Protocols for secure computations. In: Proceedings of the IEEE FOCS 1982, pp. 160–164 (1982)
4. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: Proceedings of the ACM STOC, pp. 218–229 (1987)
5. 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)
6. Raiciu, C., Rosenblum, D.S.: Enabling confidentiality in content-based publish/subscribe infrastructures. In: Proceedings of the SecureComm 2006, pp. 1–11 (2006)
7. Minami, K., Lee, A.J., Winslett, M., Borisov, N.: Secure aggregation in a publish/subscribe system. In: Proceedings of the WPES 2008, pp. 95–104 (2008)
8. Shikfa, A., Onen, 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)
9. Tariq, M.A., Koldehofe, B., Altaweel, A., Rothermel, K.: Providing basic security mechanisms in broker-less publish/subscribe systems. In: Proceedings of the ACM DEBS, pp. 38–49 (2010)
10. 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)
11. 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)

12. Di Crescenzo, G., Burns, J., Coan, B., Schultz, J., Stanton, J., Tsang, S., Wright, R.N.: Efficient and private three-party publish/subscribe. In: Lopez, J., Huang, X., Sandhu, R. (eds.) NSS 2013 LNCS, vol. 7873, pp. 278–292. Springer, Heidelberg (2013)
13. Beaver, D.: Precomputing oblivious transfer. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 97–109. Springer, Heidelberg (1995)
14. Di Crescenzo, G.: Private selective payment protocols. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 72–89. Springer, Heidelberg (2001)
15. Di Crescenzo, G.: Privacy for the stock market. In: Syverson, P.F. (ed.) FC 2001. LNCS, vol. 2339, pp. 259–278. Springer, Heidelberg (2002)
16. Aiello, W., Ishai, Y., Reingold, O.: Priced oblivious transfer: how to sell digital goods. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 119–135. Springer, Heidelberg (2001)
17. Lipmaa, H.: Verifiable homomorphic oblivious transfer and private equality test. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 416–433. Springer, Heidelberg (2003)
18. Michael, O.: Rabin: How to exchange secrets with oblivious transfer. Technical report TR-81, Aiken Computation Lab, Harvard University (1981)
19. Moni, N., Pinkas, B.: Efficient oblivious transfer protocols. In: Proceedings of the SODA 2001, pp. 448–457 (2001)
20. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Trans. Inf. Theory **22**(6), 644–654 (1976)
21. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inf. Theory **31**(4), 469–472 (1985)