2014

Towards a cryptographic treatment of publish/subscribe systems

Tsz Hon Yuen
The University of Hong Kong, thy738@uow.edu.au

Willy Susilo
University of Wollongong, wsusilo@uow.edu.au

Yi Mu
University of Wollongong, ymu@uow.edu.au

Publication Details
Yuen, T. Hon., Susilo, W. & Mu, Y. (2014). Towards a cryptographic treatment of publish/subscribe systems. Journal of Computer Security, 22 (1), 33-67.

Towards a cryptographic treatment of publish/subscribe systems

Abstract
Publish/subscribe mechanism is a typical many-to-many messaging paradigm when multiple applications want to receive the same message or when a group of applications would like to notify each other. Nonetheless, there exist only a few works that address the security issues for content-based publish/subscribe systems formally. Although the security requirements have been partially addressed by Wang et al., there is no formal definition for all of these security requirements in the literature. As a result, most of the existing schemes do not have any security proof and it is difficult to justify whether those schemes are really secure in practice. Furthermore, there is no comprehensive scheme that satisfies the most essential security requirements at the same time. In this paper, we introduce the first security model for important security requirements of content-based publish/subscribe systems. We also give a new security requirement for publisher authenticity, which means that the publisher is authenticated to publish certain types of notification only, and cannot publish other types of notification. We then exhibit a new scheme which fulfills most of the security requirements. Furthermore, we also provide a comprehensive proof for our concrete construction according to the new model. 2014 IOS Press.

Keywords
systems, cryptographic, treatment, publish, subscribe, towards

Disciplines
Engineering | Science and Technology Studies

Publication Details
Yuen, T. Hon., Susilo, W. & Mu, Y. (2014). Towards a cryptographic treatment of publish/subscribe systems. Journal of Computer Security, 22 (1), 33-67.

Towards a cryptographic treatment of publish/subscribe systems [1]

Tsz Hon Yuen [a,*], Willy Susilo [b] and Yi Mu [b]
[a] *Department of Computer Science, The University of Hong Kong, Hong Kong*
E-mail: thyuen@cs.hku.hk
[b] *Center for Computer and Information Security Research, School of Computer Science and Software Engineering, University of Wollongong, Wollongong, Australia*
E-mails: {wsusilo, ymu}@uow.edu.au

Publish/subscribe mechanism is a typical many-to-many messaging paradigm when multiple applications want to receive the same message or when a group of applications would like to notify each other. Nonetheless, there exist only a few works that address the security issues for content-based publish/subscribe systems formally. Although the security requirements have been partially addressed by Wang et al., there is *no* formal definition for all of these security requirements in the literature. As a result, most of the existing schemes do not have any security proof and it is difficult to justify whether those schemes are really secure in practice. Furthermore, there is no comprehensive scheme that satisfies the most essential security requirements at the same time. In this paper, we introduce the *first* security model for important security requirements of content-based publish/subscribe systems. We also give a new security requirement for *publisher authenticity*, which means that the publisher is authenticated to publish certain types of notification only, and cannot publish other types of notification. We then exhibit a new scheme which fulfills most of the security requirements. Furthermore, we also provide a comprehensive proof for our concrete construction according to the new model.

Keywords: Publish/subscribe, cryptographic security model, security proofs

1. Introduction

Publish/subscribe (pub/sub) is an efficient communication infrastructure that supports dynamic, many-to-many data dissemination in a distributed environment. It allows decoupled messaging between: (1) *subscribers*, having *subscriptions* to the interested information, and (2) *publishers*, providing *notifications* for the information they provide. This kind of many-to-many communication has become very popular in social networking websites.

All pub/sub technologies use subject or topic names as the loosely coupled link between publishers and subscriber systems. Publishers produce messages on a particular subject or topic name and subscribers receive those messages by registering

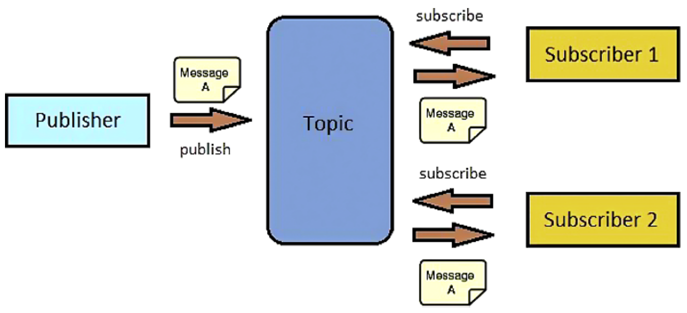

Fig. 1. Publish/subscribe system. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/JCS-130486.)

interest in the subject name either explicitly or through some broader subscription scheme using wildcards. Subscribers and publishers are loosely coupled by a network of *brokers* that route the notifications to the interested subscribers. Pub/sub allows subscribing applications to select messages by topic (as specified by the publishing application) or by content (by specifying filters). The latter is usually referred to as the content-based pub/sub systems (CBPS). Any messages addressed to a topic are delivered to all the topic's subscribers. Every subscriber receives a copy of each message. Information is automatically *pushed* to subscribing applications without having them to *pull* or request it. In short, pub/sub topologies publish messages directly to the bus or network, and these topologies are known as *shared bus*-based solutions. It is illustrated in Fig. 1.

The existing pub/sub systems tend to focus on the performance, scalability and expressiveness issues of the mechanism. Security issues and requirements are firstly addressed by Wang et al. [27]. The main issues include authentication, integrity and anonymity, which can usually be achieved by minor modification to the existing approaches. On the other hand, confidentiality is more difficult to achieve. Therefore, we consider Wang et al.'s work as addressing the security of pub/sub network *only partially*.

Our contributions. Publish/subscribe systems are important to the future social networking services. However, there is *no* formal security model for the pub/sub systems. Wang et al. [27] proposed some security issues and requirements, *without* defining a formal model. Nikander and Giannis [18] pointed out the difficulty of modeling pub/sub systems using traditional send/receive paradigm. They only gave a general model to reflect the multicast nature of the pub/sub systems, without concerning the security requirements. As a result, most of the existing schemes do not have any security theorem or proof. To the best of the authors' knowledge, only Raiciu and Rosenblum [22] proposed the security model for confidentiality and they proved the confidentiality of their scheme. A complete security theorem and proof are essential to analyze the security level of a CBPS protocol. Moreover, a complete security model is needed to identify the security requirements and the attacker's capability. Therefore, in this paper, we propose a formal security model for all security requirements for the CBPS.

Secondly, Wang et al. [27] suggested some possible solutions for each security requirements that they proposed. However, it is not clear that if these methods can work together under the *same* threat model. Moreover, some methods are out-of-band solutions and are handled independently of the pub/sub infrastructure. Additionally, most of the existing CBPS schemes enabling confidentiality do not consider authenticity and integrity simultaneously. In this paper, we propose a comprehensive CBPS scheme which fulfills most security requirements concurrently. We prove the security of our scheme under the new security model.

New improvements and comparison to our previous work [30]. Compared to our preliminary version in [30], we have made a few improvements:

- The security model of [30] did not consider the attack from a dishonest publisher, who publish notifications beyond the limitation. For example, the publisher Alice is only allowed to publish about "business news". Then she should not be able to publish about "sports news". The construction in [30] did not have any restriction about what can be published by the publisher.
  In this paper, we propose a new *publisher authenticity* security model, such that the publisher can only publish notifications within the limit imposed by the manager. Our concrete construction uses identity-based signatures to ensure that the publisher is authenticated to make such a notification.
  With the introduction of the identity-based signatures in the scheme, most security proofs has to be modified from [30]. In particular, the proofs of Theorems 3 and 5 have to consider the extra case of the forgery of the identity-based signatures.
- We provide an extension to provide subscriber confidentiality. In the basic construction in [30], the subscriber sends his request in plaintext. It may not be desirable if someone wants to keep his/her subscription in secret. In this paper, we propose an extension which provides subscriber confidentiality, by making a non-trivial integration of *public key encryption with keyword search* into our basic construction.
- We describe how to achieve publisher and subscriber anonymity from the construction in [30], by using either ring signatures or designated verifier signatures.

Related works. The Ciphertext-Policy Attribute-based Encryption (CP-ABE) is a closely related field to the pub/sub system. During encryption, the data provider can express how he wants to share data in the encryption algorithm. In traditional public key encryption, the data provider uses the recipient's public key to encrypt, such that the data is shared with the intended recipient only. In CP-ABE, the recipient is ascribed a secret key associated with a set of attributes. The data provider will provide a formula over these attributes, describing how he wants to share the data. The recipient can correctly decrypt a ciphertext encrypted with a formula only if his secret key associates with attributes which satisfy the formula. The concept of CP-ABE was firstly proposed in [23]. Several constructions of selectively secure CP-ABE systems followed, such as [2,8,9,28]. Fully secure constructions were recently provided by [13,14,19,20].

Recent works. Since the preliminary version of our work was published in [30], there have been a few related works. Some security designs for pub/sub network architecture was proposed in [12,26], but they did not have formal security proofs.

2. Definition of publish/subscribe systems

In this section we first give the definition of publish/subscribe system.

2.1. Publish/subscribe systems

A publish/subscribe system is a system with interactions between four classes of parties, as shown in Fig. 2.

- *Publishers notify* the brokers for the information they provide in the pub/sub system. They do not know who will obtain the information.
- *Subscribers subscribe* to the interested information. They only receive the information which matches their subscription.
- *Brokers match* the subscription and the notification by the subscribers and the publishers. The broker network will route and forward the packets to the matching subscribers. Sometimes they are further categorized into:
  - *Intermediate brokers.* They only route packets within the broker network.
  - *Border brokers.* They act as a link between the broker network and the other parties in the pub/sub network.

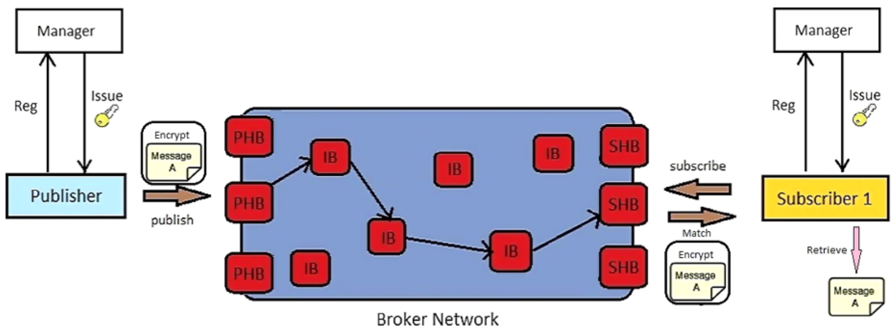

Fig. 2. Brokers in publish/subscribe system. PHB stands for Publisher Hosting Brokers, SHB stands for Subscriber Hosting Brokers and IB stands for Intermediate Brokers. PHB and SHB are both border brokers. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/JCS-130486.)

  - *Publisher hosting brokers.* They are a kind of border brokers that connect between the broker network and the publishers.
  - *Subscriber hosting brokers.* They are a kind of border brokers that connect between the broker network and the subscribers.

  Brokers are defined into these categories because they are given different level of trust in the security model. Refer to Section 2.4 for details.
- *Managers* maintain and coordinate the keys used within the pub/sub system. According to the basic concept of pub/sub system, the publisher does not know the public keys of subscribers. Therefore, the publisher cannot encrypt using subscribers' public keys. In order to provide confidentiality, managers are needed to act as the target of the encryption scheme. If confidentiality is not considered in the pub/sub system, then this party can be ignored. Some papers [16,22] assume that the publishers and the subscribers have a pre-shared session key. They do not concern about how the managers help to share the session key within the pub/sub system. The managers are called *key distribution center* in [24], *accounting server* in [10] and *secure administrator* in [32].

A content-based publish/subscribe system consists of ten algorithms defined as follows:

- $\mathsf{Setup}(1^\lambda)$: On input a security parameter 1^λ, it outputs the system parameter param and the manager's secret key msk.
- $\mathsf{KeyGen}(\mathsf{param})$: On input the system parameter param, it outputs a secret key and a public key. It can be further divided into:

  - KeyGen_p: every publisher generates his publisher secret key psk and public key ppk;
  - KeyGen_s: every subscriber generates his subscriber secret key ssk and public key spk;
  - KeyGen_b: every broker generates his broker secret key bsk and public key bpk.

- $\mathsf{RegP}(\mathsf{param}, \mathsf{filter}, \mathsf{psk})$, $\mathsf{IssueP}(\mathsf{param}, \mathsf{msk}, \mathsf{ppk})$: The interactive algorithms RegP and IssueP are run by the publisher and the manager respectively. The publisher wants to register to the manager and the manager issues a publisher key for him. The input param is the system parameter, filter is the filter set[2] by the publisher, $(\mathsf{psk}, \mathsf{ppk})$ is the publisher's secret key, public key pairs and msk is the manager's secret key. RegP first sends the filter set to IssueP and IssueP returns a publisher key K_p to RegP.
- $\mathsf{RegS}(\mathsf{param}, \mathsf{sub}, \mathsf{ssk})$, $\mathsf{IssueS}(\mathsf{param}, \mathsf{msk}, \mathsf{spk})$: The interactive algorithms RegS and IssueS are run by the subscriber and the manager respectively. The

[2]The terms filter set, subscription and notification will be explained in details later in this section.

subscriber wants to register to the manager and the manager issues a subscriber key for him. The input param is the system parameter, sub is the subscription by the subscriber, $(\mathsf{ssk}, \mathsf{spk})$ is the subscriber's secret key, public key pairs and msk is the manager's secret key. RegS first sends the subscription to IssueS and IssueS returns a subscriber key K_s to RegS.

- $\mathsf{Pub}(\mathsf{param}, m, \mathbb{A}, \mathsf{psk}, K_p)$: On input $(\mathsf{param}, m, \mathbb{A}, \mathsf{psk}, K_p)$ where param is the system parameter, m is the message, $\mathbb{A}$ is the access structure, psk is the publisher's secret key and K_p is the publisher key for some filter, the publisher outputs a notification n to the broker network. The notification n includes its encrypted content n_c and some access policy n_p to facilitate routing.
- $\mathsf{Sub}(\mathsf{param}, \mathsf{sub}, \mathsf{ssk}, \mathsf{bpk})$: On input $(\mathsf{param}, \mathsf{sub}, \mathsf{ssk}, \mathsf{bpk})$ where param is the system parameter, sub is the subscription, ssk is the subscriber's secret key and bpk is the (subscriber hosting) broker public key, the subscriber outputs a ciphertext for subscription C_{sub} to the broker network.
- $\mathsf{Match}(\mathsf{param}, n, C_{\mathsf{sub}}, \mathsf{bsk})$: On input $(\mathsf{param}, n, C_{\mathsf{sub}})$ where param is the system parameter, n is the notification, C_{sub} is the subscription ciphertext and bsk is the (subscriber hosting) broker secret key, the broker outputs spk[3] for matching the subscription, 0 for not matching, $\perp_p$ for invalid notification or $\perp_s$ for invalid subscription.
- $\mathsf{Retrieve}(\mathsf{param}, n, K_s)$: On input (param, n, K_s) where param is the system parameter, n is the notification and K_s is the subscriber key, the subscriber outputs a pair (m, ppk) or $\perp$ for invalid, where m is the message and ppk is the publisher's public key.

2.2. *Filter set, subscription and notification*

We first review the definition of monotone access structure in [1].

Monotone access structure [1]. Let $\{P_1, P_2, \ldots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1, P_2, \ldots, P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \in \mathbb{A}$. A monotone access structure is a monotone collection $\mathbb{A}$ of non-empty subsets of $\{P_1, P_2, \ldots, P_n\}$. The sets in $\mathbb{A}$ are called the *authorized sets*, and the sets not in $\mathbb{A}$ are called the *unauthorized sets*.

In our context, the subscription condition sub is treated as a set of attributes. Thus the access structure $\mathbb{A}$ will contain the authorized set of attributes (subscription condition). If the subscription condition sub satisfies the access structure $\mathbb{A}$, we write $\mathsf{sub} \in \mathbb{A}$. The filter set filter is also expressed as monotone access structure. From now on, unless stated otherwise, by an access structure we mean a monotone access structure.

[3] In practice, the broker should obtain the routing information to the subscriber spk.

2.3. Correctness

The content-based publish/subscribe system has two types of correctness: matching correctness and retrieval correctness. Matching correctness means that an honest broker can always correctly match a valid notification to a subscriber with a valid satisfying subscription. Retrieval correctness means that an honest subscriber can obtain the message if the notification which matches his subscription criteria.

Formally, they are defined as follows. For all $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda), (\mathsf{ssk}, \mathsf{spk}) \leftarrow \mathsf{KeyGen}_s(\mathsf{param})$, for all messages m and filters filter, we have:

- *Matching correctness*. We require that

  $$\mathsf{Match}\big(\mathsf{param}, n, \mathsf{Sub}(\mathsf{param}, \mathsf{sub}, \mathsf{ssk}, \mathsf{bpk}), \mathsf{bsk}\big) = \mathsf{spk},$$

  where $(\mathsf{bsk}, \mathsf{bpk}) \leftarrow \mathsf{KeyGen}(\mathsf{param}), n \leftarrow \mathsf{Pub}(\mathsf{param}, m, \mathbb{A}, \mathsf{psk}, \mathsf{RegP}(\mathsf{param}, \mathsf{filter}, \mathsf{psk})), \mathbb{A} \subseteq \mathsf{filter}$ and $\mathsf{sub} \in \mathbb{A}$.
- *Retrieval correctness*. We require that

  $$\mathsf{Retrieve}\big(\mathsf{param}, n, \mathsf{RegS}(\mathsf{param}, \mathsf{sub}, \mathsf{ssk})\big) = (m, \mathsf{ppk}),$$

  where $(\mathsf{psk}, \mathsf{ppk}) \leftarrow \mathsf{KeyGen}_p(\mathsf{param})$, $n \leftarrow \mathsf{Pub}(\mathsf{param}, m, \mathbb{A}, \mathsf{psk}, \mathsf{RegP}(\mathsf{param}, \mathsf{filter}, \mathsf{psk}))$, $\mathsf{sub} \in \mathbb{A}$ and $\mathbb{A} \subseteq \mathsf{filter}$.

2.4. Trust model

There are three types of trust regarding the underlying broker network:

(1) *A complete trust to the broker network.* The adversary is not given any information within the broker network.
(2) *A trust to the border brokers only.* The adversary can access any information in the intermediate brokers, but not the border brokers.
(3) *Untrusted broker network.* The adversary can access any information in the broker network.

Notice that the trust we discuss here is whether the brokers' keys (if any) and data accessed by the brokers are available to the adversary. We always assume that the brokers honestly route the packets. If not, the subscribers may never receive any packets.

According to different trust level of the broker network, the broker public keys and secret keys can be used as the input in the Pub, Sub or Match algorithms.

3. Security model of pub/sub systems

In this section, we define the security models for confidentiality, unforgeability and anonymity, which are the security requirements mentioned by Wang et al. [27]. We also introduce a new security requirement and model of publisher authenticity.

3.1. Confidentiality

The publish/subscribe system has three types of confidentiality: information confidentiality, subscription confidentiality and publisher confidentiality as discussed by Wang et al. [27].

Wang et al. [27] considers the case where all the contents of the notifications and subscriptions are confidential. In some cases, part of the information n_p can be known by the brokers to facilitate routing (e.g., stock code, update date in a pub-sub stock quote application) while part of the information n_c must be kept secret from untrusted brokers (e.g., stock price, percentage change). Therefore we consider the confidentiality for the secret information n_c instead of the whole document to be sent. It is easy to convert our security model into Wang et al.'s model [27]. It can be done by setting the challenge message to be a complete notification or subscription instead of a partial one.

Information confidentiality means that the secret information in the notification should not be known by the untrusted brokers and all outsiders. Subscription confidentiality means that the secret information in the subscription should not be known by the untrusted brokers, publishers, other subscribers and all outsiders. Publisher confidentiality means that the secret information in the notification should not be known by the non-subscribers of that notification. It includes the untrusted brokers and all outsiders. Therefore publisher confidentiality implies information confidentiality.

We note that our model for confidentiality only involves one trusted manager only. In real system, there may be many managers. Our model can be modified for multiple managers. We give the current confidentiality model of one manager for simplicity.

Publisher confidentiality. We describe the publisher confidentiality for the secret information in the pub/sub network. The indistinguishability game is formally defined as follows:

(1) The challenger runs $(\mathsf{param}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $(\mathsf{bsk}, \mathsf{bpk}) \leftarrow \mathsf{KeyGen}_b(\mathsf{param})$. The challenger gives the public parameters param and the secret/public key pairs of the untrusted brokers to the adversary $\mathcal{A}$. The manager's secret key msk is unknown to $\mathcal{A}$.

(2) $\mathcal{A}$ is allowed to query the following oracles:

- IssueS Oracle: On input the subscription sub and the subscriber's public key spk, it runs the $\mathsf{IssueS}(\mathsf{param}, \mathsf{msk}, \mathsf{spk})$ protocol and interacts with the $\mathsf{RegS}(\mathsf{param}, \mathsf{sub}, \cdot)$ run by $\mathcal{A}$. The oracle outputs the subscriber key K_s from IssueS.
- IssueP Oracle: On input the publisher's filter filter and the publisher's public key ppk, it runs the $\mathsf{IssueP}(\mathsf{param}, \mathsf{msk}, \mathsf{ppk})$ protocol and interacts with the $\mathsf{RegP}(\mathsf{param}, \mathsf{filter}, \cdot)$ run by $\mathcal{A}$. The oracle outputs the publisher key K_p from IssueP.

- $\mathsf{Retrieval\ Oracle}$: On input (n, sub) where n is the notification and sub is the subscription, the oracle first runs $(\mathsf{ssk}, \mathsf{spk}) \leftarrow \mathsf{KeyGen}_s(\mathsf{param})$. Then the oracle runs both $\mathsf{IssueS}(\mathsf{param}, \mathsf{msk}, \mathsf{spk})$ and $\mathsf{RegS}(\mathsf{param}, \mathsf{sub}, \mathsf{ssk})$ by itself and obtains K_s. Finally, it outputs the secret information $(m, \mathsf{ppk})/\perp \leftarrow \mathsf{Retrieve}(\mathsf{param}, n, K_s)$.

(3) $\mathcal{A}$ sends two messages m_0^* and m_1^* from the message space, a publisher secret key psk^*, a filter filter^* and an access structure $\mathbb{A}^* \subseteq \mathsf{filter}^*$ to the challenger. The challenger encrypts m_b^* as $n^* \leftarrow \mathsf{Pub}(\mathsf{param}, m_b^*, \mathbb{A}^*, \mathsf{psk}^*, \mathsf{RegP}(\mathsf{param}, \mathsf{filter}^*, \mathsf{psk}^*))$. There should be no subscription sub queried to the $\mathsf{IssueS\ Oracle}$, such that $\mathsf{sub} \in \mathbb{A}^*$. The challenger picks a bit $b \in \{0, 1\}$ and sends the notification n_b^* to $\mathcal{A}$.

(4) $\mathcal{A}$ is allowed to query the oracles, with the exception that no subscription sub queried to the $\mathsf{IssueS\ Oracle}$, such that $\mathsf{sub} \in \mathbb{A}^*$; and n^* should not be queried to the $\mathsf{Retrieval\ Oracle}$.

(5) Finally $\mathcal{A}$ output his guess b'.

The advantage of $\mathcal{A}$ in the game is $|\Pr[b' = b] - \frac{1}{2}|$.

Definition. A CBPS scheme is $(\varepsilon, t, q_s, q_p, q_r)$-publisher confidential against chosen ciphertext attack if there is no t-time adversary with q_s queries to the IssueS oracle, q_p queries to the IssueP oracle and q_r queries to the Retrieval oracle has an advantage over ε in the game.

Information confidentiality. Due to the similarity of the definition between information confidentiality and publisher confidentiality, we can define the indistinguishability game of publisher confidentiality same as the one of information confidentiality without query to the IssueS Oracle.

Definition. A CBPS scheme is $(\varepsilon, t, q_p, q_r)$-information confidential against chosen ciphertext attack if there is no t-time adversary with q_p queries to the IssueP oracle and q_r queries to the retrieval oracle has an advantage over ε in the game.

Notice that for both publisher and information confidentiality, we say that a system is *selectively* secure if we require the adversary commits to the challenge filter filter^* at the beginning of the game.

Subscription confidentiality. The subscribers may want their subscriptions to be confidential against the broker network.[4] Then the brokers need to match the "encrypted subscriptions" with the notifications.[5] The indistinguishability game is defined as follows:

[4]For example, an investor may not want other people to know which stock price he has subscribed, since it may leak information of which stock he may buy.

[5]Public key Encryption with Keyword Search [3] can be used to solve this dilemma. We will explain in details in Section 6.

(1) The challenger runs $(\mathsf{param}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $(\mathsf{bsk}, \mathsf{bpk}) \leftarrow \mathsf{KeyGen}_b(\mathsf{param})$. The challenger gives the public parameters param and the secret/public key pairs of the untrusted brokers to the adversary $\mathcal{A}$. The manager's secret key msk is unknown to $\mathcal{A}$.
(2) $\mathcal{A}$ is allowed to query the IssueS Oracle, IssueP Oracle and Retrieval Oracle defined in the publisher confidentiality game.
(3) $\mathcal{A}$ sends two subscription sub_0^* and sub_1^* and the subscriber secret key ssk^*, where sub_0^* and sub_1^* have never been queried to the IssueS Oracle. The challenger picks a bit $b \in \{0, 1\}$ and computes $C_{\mathsf{sub}}^* \leftarrow \mathsf{Sub}(\mathsf{param}, \mathsf{sub}_b^*, \mathsf{ssk}^*, \mathsf{bpk})$. He sends the resulting ciphertext C_{sub}^* to $\mathcal{A}$.
(4) $\mathcal{A}$ is allowed to query the oracles, with the exception that no subscription sub_0^* and sub_1^* are queried to the IssueS Oracle.
(5) Finally $\mathcal{A}$ output his guess b'.

The advantage of $\mathcal{A}$ in the game is $|\Pr[b' = b] - \frac{1}{2}|$.

Definition. A CBPS scheme is $(\varepsilon, t, q_s, q_p, q_r)$-subscription confidential against chosen ciphertext attack if there is no t-time adversary with q_s queries to the IssueS oracle, q_p queries to the IssueP oracle and q_r queries to the Retrieval oracle has an advantage over ε in the game.

Notice that all of the above definitions for confidentiality is against chosen ciphertext attack (CCA). If we do not allow any query to the retrieval oracle, then the above confidentiality definition is reduced to against chosen plaintext attack (CPA).

3.2. Unforgeability

Unforgeability provides authentication and integrity for the pub/sub system. Wang et al. [27] mentioned that authentication (end-to-end and point-to-point), information integrity, subscription integrity and service integrity are important security requirements for the pub/sub system. We use the standard notion of unforgeability for digital signature to cover the authentication and integrity requirements.

Information unforgeability. Information unforgeability means that the subscriber believes that the notification is produced by the publisher and is not altered in the broker network. The game for information unforgeability is formally defined as follows:

(1) The challenger runs $(\mathsf{param}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{bsk}, \mathsf{bpk}) \leftarrow \mathsf{KeyGen}_b(\mathsf{param})$ and $(\mathsf{psk}, \mathsf{ppk}) \leftarrow \mathsf{KeyGen}_p(\mathsf{param})$. The challenger gives the public parameters param, the manager's secret key msk, the secret/public key pairs of the untrusted brokers and the publisher public key ppk to the adversary $\mathcal{A}$. The publisher's secret key psk is unknown to $\mathcal{A}$.

(2) $\mathcal{A}$ is allowed to query the $\mathsf{Pub\ Oracle}$: On input a message m, the publisher filter filter and an access structure $\mathbb{A} \subseteq \mathsf{filter}$, the oracle first runs both $\mathsf{IssueP}(\mathsf{param}, \mathsf{msk}, \mathsf{ppk})$ and $\mathsf{RegP}(\mathsf{param}, \mathsf{filter}, \mathsf{psk})$ by itself and obtains K_p. Then, it outputs the notification $n \leftarrow \mathsf{Pub}(\mathsf{param}, m, \mathbb{A}, \mathsf{psk}, K_p)$.
(3) $\mathcal{A}$ returns a message m^*, a notification n^* for a subscription sub^*.

$\mathcal{A}$ wins if $(m^*, \mathsf{ppk}) \leftarrow \mathsf{Retrieve}(\mathsf{param}, n^*, K_s)$, where K_s is the output of $\mathsf{RegS}(\mathsf{param}, \mathsf{sub}^*, \mathsf{ssk}))$ interacting with $\mathsf{IssueS}(\mathsf{param}, \mathsf{msk}, \mathsf{spk})$, n^* was not the output of $\mathsf{Pub\ Oracle}$ query with input m^* and $(\mathsf{ssk}, \mathsf{spk}) \leftarrow \mathsf{KeyGen}_s(\mathsf{param})$.

Definition. A CBPS scheme is $(\varepsilon,\ t,\ q_p)$-information unforgeable against chosen message attack if there is no t-time adversary winning the above game with probability at least ε with q_p queries to the Pub oracle.

Subscription unforgeability. Subscription unforgeability means that the broker believes that the subscription is produced by the subscriber and is not altered in the broker network. The game for subscription unforgeability is formally defined as follows:

(1) The challenger runs $(\mathsf{param}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{bsk}, \mathsf{bpk}) \leftarrow \mathsf{KeyGen}_b(\mathsf{param})$ and $(\mathsf{ssk}, \mathsf{spk}) \leftarrow \mathsf{KeyGen}_s(\mathsf{param})$. The challenger gives the public parameters param, the manager's secret key msk, the secret/public key pairs of the untrusted brokers and the subscriber's public key spk to the adversary $\mathcal{A}$. The subscriber's secret key ssk is unknown to $\mathcal{A}$.
(2) $\mathcal{A}$ is allowed to query the $\mathsf{Sub\ Oracle}$: On input the subscription sub, the oracle first runs both $\mathsf{IssueS}(\mathsf{param}, \mathsf{msk}, \mathsf{spk})$ and $\mathsf{RegS}(\mathsf{param}, \mathsf{sub}, \mathsf{ssk})$ by itself and obtains K_s. Then, it outputs the subscription ciphertext $C_{\mathsf{sub}} \leftarrow \mathsf{Sub}(\mathsf{param}, \mathsf{sub}, \mathsf{ssk}, \mathsf{bpk})$.
(3) $\mathcal{A}$ returns a subscription ciphertext C^*_{sub} and a notification n^*.

$\mathcal{A}$ wins if $\mathsf{spk} \leftarrow \mathsf{Match}(\mathsf{param}, n^*, C^*_{\mathsf{sub}}, \mathsf{bsk})$ and C^*_{sub} was not the output of $\mathsf{Sub\ Oracle}$ query.

Definition. A CBPS scheme is $(\varepsilon,\ t,\ q_s)$-subscription unforgeable against chosen message attack if there is no t-time adversary winning the above game with probability at least ε with q_s queries to the Sub oracle.

Service unforgeability. Service unforgeability means that the broker believes that the notification is produced by the publisher and is not altered in the previous broker network. It ensures that once malicious faults arises at the infrastructure level, it could be detected by the next broker. Information unforgeability provides end-to-end authentication of the publisher, while service unforgeability provides authentication of the publisher to every point in the network. It minimizes the damage by a malicious broker who insert bogus notifications into the pub/sub network. The game for information unforgeability is formally defined as follows:

(1) It is the same as the first step of the game for information unforgeability.
(2) $\mathcal{A}$ is allowed to query the $\textsf{Pub Oracle}$ defined in the information unforgeability game.
(3) $\mathcal{A}$ returns a notification n^*, a subscription ciphertext $C^*_{\textsf{sub}}$, a subscriber's public key $\textsf{spk}^*$ and the corresponding subscriber key K^*_s.

$\mathcal{A}$ wins the game if $(m^*, \textsf{ppk}) \leftarrow \textsf{Retrieve}(\textsf{param}, n^*, K^*_s)$, $\textsf{spk}^* \leftarrow \textsf{Match}(\textsf{param}, n^*, C^*_{\textsf{sub}}, \textsf{bsk})$ and n^* was not the output of any $\textsf{Pub Oracle}$ query.

Definition. A CBPS scheme is (ε, t, q_p)-service unforgeable against chosen message attack if there is no t-time adversary winning the above game with probability at least ε with q_p queries to the Pub oracle.

3.3. Anonymity

The anonymity in the pub/sub system is different for the publishers and subscribers. We will consider two cases separately. The trust model for anonymity is slightly different from confidentiality and unforgeability, since the border brokers directly connecting to the publisher and subscriber must know who is communicating with them. The border brokers know the IP address or the MAC address of the publisher and subscriber for routing purpose (although their public keys or identities may still be hidden). Therefore the border brokers should be trusted for anonymity related to connectivity (e.g., an IP address). To be more specific, publisher hosting broker is trusted for publisher anonymity; and subscriber hosting broker is trusted for subscriber anonymity. On the other hand, the public keys of the publisher and subscriber may still be anonymous to all brokers, if the IP/MAC address is not related to the public keys.

Publisher anonymity. The anonymity for the publisher means the publisher remains anonymous when he sends a notification. Only the legitimate subscribers can know the identity of the publisher (for authentication purpose). The publisher anonymity game is formally defined as follows:

(1) The challenger runs $(\textsf{param}, \textsf{msk}) \leftarrow \textsf{Setup}(1^\lambda)$ and $(\textsf{bsk}, \textsf{bpk}) \leftarrow \textsf{KeyGen}_b(\textsf{param})$. The challenger gives the public parameters $\textsf{param}$ and the secret/public key pairs of the untrusted brokers to the adversary $\mathcal{A}$. The manager's secret key $\textsf{msk}$ is unknown to $\mathcal{A}$.
(2) $\mathcal{A}$ is allowed to query the $\textsf{IssueS Oracle}$, $\textsf{IssueP Oracle}$ and $\textsf{Retrieval Oracle}$ defined in the publisher confidentiality game.
(3) $\mathcal{A}$ sends two publisher key pairs $(\textsf{ppk}^*_0, \textsf{psk}^*_0)$ and $(\textsf{ppk}^*_1, \textsf{psk}^*_1)$, a message m^* and a filter $\textsf{filter}^*$ to the challenger. The challenger picks a bit $b \in \{0, 1\}$ and computes $n^* = (n^*_c, n^*_p) \leftarrow \textsf{Pub}(\textsf{param}, m^*, \textsf{psk}^*_b, \textsf{RegP}(\textsf{param}, \textsf{filter}^*, \textsf{psk}^*_b))$. There should be no subscription $\textsf{sub}$ queried to the $\textsf{IssueS Oracle}$, such that $\textsf{sub} \in n^*_p$, no matter $b = 0$ or 1. It sends the notification n^* to $\mathcal{A}$.

(4) $\mathcal{A}$ is allowed to query the oracles, with the exception that no subscription sub queried to the $\mathsf{IssueS\ Oracle}$, such that $\mathsf{sub} \in n_p^*$; and n^* should not be queried to the $\mathsf{Retrieval\ Oracle}$.
(5) Finally $\mathcal{A}$ output his guess b'.

The advantage of $\mathcal{A}$ in the game is $|\Pr[b' = b] - \frac{1}{2}|$.

Definition. A CBPS scheme is $(\varepsilon, t, q_s, q_p, q_r)$-publisher anonymous against chosen ciphertext attack if there is no t-time adversary with q_s queries to the IssueS oracle, q_p queries to the IssueP oracle and q_r queries to the retrieval oracle has an advantage over ε in the game.

Subscriber anonymity. The anonymity for the subscriber means that the subscriber remains anonymous when he sends a subscription. The subscriber anonymity game is formally defined as follows:

(1) The challenger runs $(\mathsf{param}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $(\mathsf{bsk}, \mathsf{bpk}) \leftarrow \mathsf{KeyGen}_b(\mathsf{param})$. The challenger gives the public parameters param, the manager's secret key msk and the subscriber hosting broker's public key bpk to the adversary $\mathcal{A}$. The subscriber hosting broker's secret key bsk is unknown to $\mathcal{A}$.
(2) $\mathcal{A}$ is allowed to query the following oracles: $\mathsf{Match\ Oracle}$: On input (n, C_{sub}) where n is the notification and C_{sub} is the subscription ciphertext to bpk, it outputs the matching result: spk, 0, $\perp_s$ and/or $\perp_p$ which is the output from $\mathsf{Match}(\mathsf{param}, n, C_{\mathsf{sub}}, \mathsf{bsk})$.
(3) $\mathcal{A}$ sends two subscriber key pairs $(\mathsf{spk}_0^*, \mathsf{ssk}_0^*)$ and $(\mathsf{spk}_1^*, \mathsf{ssk}_1^*)$, a subscription sub^* to the challenger. The challenger picks a bit $b \in \{0, 1\}$ and computes $C_{\mathsf{sub}}^* \leftarrow \mathsf{Sub}(\mathsf{param}, \mathsf{sub}^*, \mathsf{ssk}_b^*, \mathsf{bpk})$. He sends the subscription ciphertext C_{sub}^* to $\mathcal{A}$.
(4) $\mathcal{A}$ is allowed to query the oracles, with the exception that no subscription ciphertext C_{sub}^* queried to the $\mathsf{Match\ Oracle}$.
(5) Finally $\mathcal{A}$ output his guess b'.

The advantage of $\mathcal{A}$ in the game is $|\Pr[b' = b] - \frac{1}{2}|$.

Definition. A CBPS scheme is (ε, t, q_m)-subscriber anonymous against chosen ciphertext attack if there is no t-time adversary with q_m queries to the Match oracle has an advantage over ε in the game.

There are a few different anonymity requirements in different applications, which may require different security models. We give a few examples here.

(1) *Accountability*: Subscription anonymity may contradict the accountability requirement in [27]. In commercial pub/sub applications, publishers may want to charge subscribers for the information they provide. If the charge is time basis, subscribers pay when they get the subscription key for a period of time from the manager. Each independent subscription can still be anonymous and our

current subscription anonymity model can still be used. However if the charge is per notification basis, subscribers' identities must be revealed for accountability and auditability purposes. The security model need to be changed, such that a publisher needs to know whose subscription matches his notification.

(2) *Full anonymity*: Our security model above implies that the legitimate subscribers know the publisher's public key ppk, and the subscriber hosting broker knows the subscriber's public key spk. For applications where full anonymity is required, anonymous credential [5] or private credential [4] schemes can be used on top of our pub/sub system. Then the ppk and spk becomes the pseudonyms for the publisher and the subscriber respectively. Even if these pseudonyms are known to the brokers and other outsiders, it will not affect the anonymity of the system. The anonymity model needs to be changed to the one similar to the anonymous credential or private credential system. The manager in CBPS also has to play the role of the trusted credential issuing party. On the other hand, we can also use attribute-based signatures [15] on top of our pub/sub system. Then the ppk and spk becomes the attributes for the publisher and the subscriber respectively. These attributes cannot reveal the real identities of the publisher and the subscriber. Therefore full anonymity may also be preserved in this case.

3.4. Publisher authenticity

Compare with [27], our pub/sub system has an extra layer of publisher authentication. The publisher can only publish notification satisfying the filter, which was defined when the publisher requested a key from the manager. Therefore, we require that a publisher cannot publish any notification that does not satisfy his filter. The publisher authenticity game is formally defined as follows:

(1) The challenger runs $(\mathsf{param}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{bsk}, \mathsf{bpk}) \leftarrow \mathsf{KeyGen}_b(\mathsf{param})$ and $(\mathsf{psk}, \mathsf{ppk}) \leftarrow \mathsf{KeyGen}_p(\mathsf{param})$. The challenger gives the public parameters param, the secret/public key pairs of the untrusted brokers and the publisher public key ppk and secret key psk to the adversary $\mathcal{A}$. The manager's secret key msk is unknown to $\mathcal{A}$.
(2) $\mathcal{A}$ is allowed to query the $\mathsf{IssueP\ Oracle}$ and the $\mathsf{Pub\ Oracle}$ as defined in publisher confidentiality and information unforgeability respectively.
(3) $\mathcal{A}$ returns a message m^*, a notification n^* for a subscription sub^*.

$\mathcal{A}$ wins if $(m^*, \mathsf{ppk}) \leftarrow \mathsf{Retrieve}(\mathsf{param}, n^*, K_s)$, where K_s is the output of $\mathsf{RegS}(\mathsf{param}, \mathsf{sub}^*, \mathsf{ssk})$ interacting with $\mathsf{IssueS}(\mathsf{param}, \mathsf{msk}, \mathsf{spk})$, $(\mathsf{ssk}, \mathsf{spk}) \leftarrow \mathsf{KeyGen}_s(\mathsf{param})$ and there is no IssueP oracle query with input $(\mathsf{filter}, \mathsf{ppk})$ such that $\mathsf{sub}^* \in \mathsf{filter}$.

Definition. A CBPS scheme is $(\varepsilon, t, q_I, q_p)$-publisher authenticity against chosen message attack if there is no t-time adversary winning the above game with probability over ε with q_I queries to the IssueP oracle and q_p queries to the Pub oracle.

4. Our construction

In this section, we will present our new construction of CBPS. We first describe the main idea of the scheme. We review the relevant cryptographic background and then we show the basic construction.

4.1. Main idea of our basic scheme

A secure CBPS protocol should have security proofs for both confidentiality and unforgeability. To provide information confidentiality and information unforgeability at the same time, we use an approach commonly used in signcryption schemes. It means that the randomness used in the signature and the encryption are the same. It ensures that the signature and the encryption protocol are run by the same party. An adversary cannot use the ciphertext from a legitimate user and append the adversary's signature to it; nor use the signature from a legitimate user and append a ciphertext computed by the adversary.

To facilitate routing while providing confidentiality in the pub/sub system, we employ the approach that only the part of the document containing the secret information is encrypted. For example, in a pub-sub stock quote application, a publisher (the bank) provides stock quote to subscribers (the bank's customers). The stock price is encrypted while the stock name is not. Therefore the document can be routed to subscribers who are interested in a particular stock.

Unforgeability. The challenge of encrypting the partial document is how the brokers authenticate the document without knowing the plaintext. Referring to the previous example, an obvious solution is to sign on the stock name and the encrypted stock price. However, a signature on the encrypted stock price does not guarantee the authenticity of the stock price. A more complicated solution in [10] is to encrypt the stock price and the signature of the stock price. After that the stock name and the whole ciphertext is signed again.

In this section, we use a simpler approach by sanitizable signatures [17]. A sanitizable signature scheme allows one to verify a signature even when part to the original message is not known. Therefore, we can compute a sanitizable signature to the whole document and encrypt the stock price. The brokers only need to verify the signature for the part of the stock name. For the subscribers, the same signature is verified against the whole document after decryption. By the property of sanitizable signatures, it is difficult to obtain the sanitized messages (i.e. the stock price) from the sanitizable signature. Therefore authenticity is preserved while having confidentiality in the (untrusted) broker network. Our scheme uses the sanitizable signatures by Suzuki et al. [25].

Confidentiality. The challenge of confidentiality is that how the publisher can restrict the access of the secret information. By the loose coupling property of the pub/sub network, the publisher does not know who subscribe the notifications. Hence the publisher has no public key to encrypt the secret information. Some schemes [16, 22] assume that the publisher and the subscriber share a symmetric key. However, it contradicts the very first assumption of decoupling of publishers and subscribers. These schemes are only suitable for private pub/sub systems over public networks. An internet-scale, dynamic pub/sub network with a universe of publishers and subscribers are unlikely to share a symmetric key. Another solution [21] for confidentiality is through access control to the broker network. Encryption and decryption is performed by the border brokers and therefore trust must be placed upon them. If the broker network is not trusted, it is difficult for the publisher to find a suitable public key for encryption.

Our scheme uses the Ciphertext-Policy Attribute-Based Encryption (CP-ABE) by Waters [28] to solve this problem. In CP-ABE, attributes are used to describe the user's (subscriber's) credentials and the encrypting party (publisher) can encrypt the message according to some formulas over these credentials. Therefore, the publisher can encrypt the secret information by some suitable attributes. Subscribers can request keys from the manager about the attributes that they are interested in.

Publisher authenticity. By using CP-APE, the encryption policy $\mathbb{A}$ is known in the ciphertext. However, the subscriber and the brokers may want to check if the publisher has obtained permission from the manager for publishing in such policy. We use identity-based signatures in [7] to provide such authentication. During the registration phase, the publisher ppk request the permit to publish to some policy belongs to filter. The manager generates an identity-based secret key to the publisher, by treating $(\mathsf{ppk}, \mathsf{filter})$ as the identity. Then, in the Pub algorithm, the notification will also include an identity-based signature on the notification content, using the identity-based secret key. Therefore, the subscriber and the brokers can check if the identity-based signature is valid with respect to the identity $(\mathsf{ppk}, \mathsf{filter})$, and $\mathbb{A} \subseteq \mathsf{filter}$.

4.2. *Cryptographic backgrounds*

We present a brief revision on groups with efficiently computable bilinear maps and then review some number theoretic assumptions. After that, we review the definition of access structures and relevant backgrounds on Linear Secret Sharing Schemes, sanitizable signatures and identity-based signatures. They are extensively used in our concrete construction of pub/sub system.

Pairings and intractability assumptions. Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}$. A map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is called a bilinear map if, for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, we have $\hat{e}(v^a, v^b) = \hat{e}(u, v)^{ab}$ and $\hat{e}(g, g) \neq 1$.

CDH. The Computational Diffie–Hellman problem is that, given $g, g^x, g^y \in \mathbb{G}$ for unknown $x, y \in \mathbb{Z}_p$, to compute g^{xy}. We say that the (ε, t)-CDH assumption holds if no t-time algorithm has the non-negligible probability ε in solving the CDH problem.

DBDH. The Decisional Bilinear Diffie–Hellman problem is that, given $(g, g^a, g^b, g^c) \in \mathbb{G}$ and $T \in \mathbb{G}_T$ for unknown $a, b, c \in \mathbb{Z}_p$, to distinguish if $T = \hat{e}(g, g)^{abc}$ or T is a random element in $\mathbb{G}_T$. We say that the (ε, t)-DBDH assumption holds if no t-time algorithm has the non-negligible probability ε minus half in solving the DBDH problem.

Decisional q-BDHE. The decisional q-Bilinear Diffie–Hellman Exponent problem is that, given $(g, g^a, g^{a^2}, \ldots, g^{a^q}, g^{a^{q+2}}, \ldots, g^{a^{2q}}, g^s) \in \mathbb{G}$ and $T \in \mathbb{G}_T$ for unknown $a, s \in \mathbb{Z}_p$, to distinguish if $T = \hat{e}(g, g)^{a^{q+1}s}$ or T is a random element in $\mathbb{G}_T$. We say that the (ε, t)-decisional q-BDHE assumption holds if no t-time algorithm has the non-negligible probability ε minus half in solving the decisional q-BDHE problem.

Linear Secret Sharing Schemes. We adapt the definition of Linear Secret Sharing Schemes (LSSS) in [1]. A secret s is shared to at most ℓ parties, out of the total n parties.

LSSS [1]. A secret sharing scheme Π over a set of parties $\mathcal{P}$ is called *linear* over $\mathbb{Z}_p$ if:

(1) The shares for each party form a vector over $\mathbb{Z}_p$.
(2) There exists a matrix M called the share-generating matrix for Π. The matrix M has ℓ rows and n columns. For $i = 1, \ldots, \ell$, the ith row of M we let the function ρ defined the party labeling row i as $\rho(i)$. When we consider the column vector $v = (s, r_2, \ldots, r_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared, and $r_2, \ldots, r_n \in \mathbb{Z}_p$ are randomly chosen, then Mv is the vector of ℓ shares of the secret s according to Π. The share $(Mv)_i$ belongs to party $\rho(i)$.

Beimel [1] showed that every LSSS enjoys the *linear reconstruction* property, defined as follows: Suppose that Π is an LSSS for the access structure $\mathbb{A}$. Let $S \in \mathbb{A}$ be any authorized set, and let $I \subset \{1, \ldots, \ell\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then there exist constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ such that, if $\{\lambda_i\}$ are valid shares of any secret s according to Π, then $\sum_{i \in I} \omega_i \lambda_i = s$. Furthermore, Beimel [1] showed that these constants $\{\omega_i\}$ can be found in time polynomial in the size of the share-generating matrix M.

Sanitizable signatures. A digital signature prohibits any alteration of the original message once it is signed. It protects the signer against the message forgery. Nevertheless, it also prevents the message from being process further legitimately as well, which sometimes is actually desirable.

For example, the government wants to release some *partial* information in an officially signed document. A government officer wants to delete some sensitive information such as personal information or national secrets. In order to avoid signing the message again (since the original signer may not be available at that time), a sanitizable signature can be used to sign the document at the first place; and the sensitive information can be sanitized prior to the release of the signature. The goal of sanitizable signatures is to protect the confidentiality of part of the document while ensuring the integrity of it.

There are a few different definitions of sanitizable signatures in the literature. A comprehensive review can be found in [29]. In this paper, we consider the following definitions for the sanitizable signatures used in this paper: Each sanitized message is represented by some special character, such that everyone can notice where the document is sanitized. The signer cannot choose who are the designated sanitizers when he signs the document.

One-time symmetric-key encryption. We use the paradigm of hybrid encryption scheme [11] to encrypt the message in our scheme. We review their definition on one-time symmetric-key encryption scheme $\mathcal{SKE}$. $\mathcal{SKE} = (\mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ consists of two polynomial-time algorithms. The encryption algorithm $\mathsf{SKE.Enc}(1^\lambda, K, m) \rightarrow C$, takes as input the security parameter 1^λ, a key K and a message m and outputs a ciphertext C; in the decryption algorithm $\mathsf{SKE.Dec}(1^\lambda, K, C) \rightarrow m$, a possessor of the key K decrypts the ciphertext C to get back a message m or the special rejection symbol $\perp$. The key K is a bit string of length $\mathsf{SKE.Len}(\lambda)$, where $\mathsf{SKE.Len}(\lambda)$ is a parameter of the encryption scheme.

Identity-based signatures. In identity-based signatures (IBS), there is a trusted authority to generate secret keys for users with different identities. The user can use the identity-based secret key to sign a message, and the verifier can verify the signature using the identity and the master public key of the trusted authority. We use the identity-based signature scheme IBS in [7] for the publisher to show that he has the suitable filter.

- $\mathsf{IBS.Setup}$: The manager, treated as the trusted authority in IBS, randomly picks his secret key $\mathsf{msk} = \beta \in \mathbb{Z}_p$ and sets his public key $\mathsf{mpk} = (g, g^\beta, h, \hat{H}_1, \hat{H}_2)$, where g, h are generators of a group $\mathbb{G}$ and $\hat{H}_1, \hat{H}_2$ are collision resistant hash functions.
- $\mathsf{IBS.Extract}$: The publisher requests an identity-based secret key K, where the identity id is the requested filter. The manager calculates $K = \hat{H}_1(\mathsf{id})^\beta$.
- $\mathsf{IBS.Sign}$: To generate a signature on a message M, the publisher picks a random number $r \in \mathbb{Z}_p$ and outputs $\sigma = (S_1, S_2)$, where $S_1 = g^r$ and $S_2 = K^{\hat{H}_2(\mathsf{id}, M)} h^r$.
- $\mathsf{IBS.Verify}$: It accepts the signature if $\hat{e}(g, S_2) = \hat{e}(g^\beta, \hat{H}_1(\mathsf{id}))^{\hat{H}_2(\mathsf{id}, M)} \cdot \hat{e}(S_1, h)$.

4.3. The basic scheme

We use the sanitizable signature scheme by Suzuki et al. [25] and CP-ABE scheme by Waters [28]. Some input parameters described in Section 2.1 are omitted here when they are not used in the basic scheme. Let $(\mathsf{S.KeyGen}, \mathsf{S.Sig}, \mathsf{S.Vfy})$ be a EUF-CMA (existentially unforgeable against chosen message attack) secure signature scheme. Let $(\mathsf{IBS.Setup}, \mathsf{IBS.Ext}, \mathsf{IBS.Sig}, \mathsf{IBS.Vfy})$ be the identity based signature scheme in [7]. Let $(\mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ be a secure one-time symmetric-key encryption scheme.

- Setup. On input 1^λ, it picks the pairing $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, generators $g, g_1 \in \mathbb{G}$ and collision resistant hash functions $H_1 : \{0,1\}^* \to \mathbb{G}$ and $H_2 : \{0,1\}^* \to \mathbb{G}$. Let $H_3 : \mathbb{G} \to \{0,1\}^k$ be a hash function, where $k = \mathsf{SKE.Len}(\lambda)$, and $H_3(v)$ is uniformly distributed over $\{0,1\}^k$ if v is uniformly distributed over $\mathbb{G}$. It chooses a random exponent $\alpha \in \mathbb{Z}_p$ and runs $(\mathsf{IBS.mpk}, \mathsf{IBS.msk}) \leftarrow \mathsf{IBS.Setup}(1^\lambda)$. The manager secret key is $(g^\alpha, \mathsf{IBS.msk})$. It outputs the system parameter $\mathsf{param} = \{g, g_1, \hat{e}(g,g)^\alpha, \hat{e}, H_1, H_2, H_3, \mathsf{IBS.mpk}\}$.
- KeyGen. On input the system parameter param, the publisher randomly picks his secret key $x_p \leftarrow \mathbb{Z}_p$. He outputs his public key $y_p = g^{x_p}$. The subscriber runs $\mathsf{S.KeyGen}(1^\lambda)$ and obtains his secret key x_s and public key y_s.
- RegP, IssueP. The publisher chooses the filter as an LSSS access structure (M, ρ). We limit ρ to be an injective function,[6] that is an attribute is associated with at most one row of an $\ell \times n$ matrix M.[7] The manager uses his secret key to run $\mathsf{sk}_m = \mathsf{IBS.Ext}(\mathsf{IBS.msk}, (y_p, M, \rho))$. The publisher key $K_p = (\mathsf{sk}_m, (M, \rho))$.
- RegS, IssueS. On input the subscription as a set of attributes S, the subscriber sends it to the manager. The manager (with master secret key g^α) chooses a random $t \in \mathbb{Z}_p$ and sends the subscriber key K_s to the subscriber, where

$$K_s = \big(K = g^\alpha g_1^t, L = g^t, \forall x \in S, K_x = H_1(x)^t\big).$$

- Pub. On input $(\mathsf{param}, m, \mathbb{A}, x_p, K_p)$ where param is the system parameter, m is the message, $\mathbb{A} = (M, \rho)$ is the access structure, x_p is the publisher's secret key and $K_p = (\mathsf{sk}_m, (M', \rho'))$ is the publisher key. Suppose the access structure $(M, \rho) \subset (M', \rho')$. It first chooses a random vector $\vec{v} = (s, y_2, \ldots, y_n) \in \mathbb{Z}_p^n$.[8] For $i = 1, \ldots, \ell$, he calculates $\lambda_i = \vec{v} \cdot M_i$, where M_i is the vector corresponding to the ith row of M. The publisher then chooses random $r_1, r_2 \in \mathbb{G}$ and $s \in \mathbb{Z}_p$ and computes

$$w_1 = H_2(m \parallel r_1), \qquad w_2 = H_2(M \parallel \rho \parallel r_2),$$
$$C' = g^s, \qquad C = \mathsf{SKE.Enc}\big(1^\lambda, H_3\big(\hat{e}(g,g)^{\alpha s}\big), (m, \sigma_1, r_1)\big),$$

[6]This restriction is crucial to the security proof. As in [13], such system is called as a One-Use system. We can use the encoding technique in [13] to extend it to a Multi-Use system.

[7]n is the number of possible attributes in the system. The subscriber is required to have a certain number of matching attributes in order to retrieve the notification, which is defined by ρ. ℓ is the maximum number of matching attributes.

[8]The secret value s is used to hide the message m in the ciphertext C. The values $y_2, \ldots, y_n$ are used as the random binding values as in the LSSS scheme.

$$C_1 = g_1^{\lambda_1} H_1\big(\rho(1)\big)^{-s}, \ldots, \qquad C_\ell = g_1^{\lambda_\ell} H_1\big(\rho(\ell)\big)^{-s},$$
$$\sigma_1 = w_1^{x_p}, \qquad w_3 = H_2\big(w_1 \parallel w_2 \parallel C' \parallel C \parallel C_1 \parallel \cdots \parallel C_\ell\big),$$
$$\sigma_2 = (w_2 w_3)^{x_p}.$$

The notification content is $n_c = (C', r_2, w_1, \sigma_2, C, C_1, \ldots, C_\ell)$.[9] It computes $\sigma_n = (S_1, S_2) \leftarrow \mathsf{IBS.Sign}(\mathsf{sk}_m, (n_c, M, \rho))$ using the same randomness s. Therefore $S_1 = C'$. It sets the notification policy as $n_p = ((M, \rho), y_p, \sigma_n, (M', \rho'))$. It publishes $n = (n_c, n_p)$.

- Sub. On input (param, x_s) where param is the system parameter and x_s is the subscriber's secret key, the subscriber signs the subscription attributes S by $\sigma_s = \mathsf{S.Sig}(x_s, S)$. He sends $C_{\mathsf{sub}} = (S, \sigma_s, y_s)$ to the broker network.
- Match. On input $(\mathsf{param}, (n_c, n_p), C_{\mathsf{sub}})$ where param is the system parameter, $n_c = (C', r_2, w_1, \sigma_2, C, C_1, \ldots, C_\ell)$, $n_p = ((M, \rho), y_p, \sigma_n, (M', \rho'))$ is the notification and $C_{\mathsf{sub}} = (S, \sigma_s, y_s)$ is the subscription, the broker first checks the correctness of n_p: (1) the publisher is authenticated with the filter (identity) (M', ρ') by checking if $\top \leftarrow \mathsf{IBS.Verify}(\mathsf{IBS.mpk}, (y_p, M', \rho'), \sigma_n, (n_c, M, \rho))$, (2) the notification satisfies the filter by checking if $(M, \rho) \subset (M', \rho')$, and (3) $S_1 = C'$. The broker outputs $\perp_p$ if either checking fails. Otherwise, it computes

$$w_2 = H_2(M \parallel \rho \parallel r_2),$$
$$w_3 = H_2\big(w_1 \parallel w_2 \parallel C' \parallel C \parallel C_1 \parallel \cdots \parallel C_\ell\big).$$

If $\hat{e}(\sigma_2, g) \neq \hat{e}(w_2 w_3, y_p)$, the broker outputs $\perp_p$. If $\mathsf{S.Vfy}(y_s, S, \sigma_s) = 0$, the broker also outputs $\perp_s$.
Otherwise, when S satisfies the access structure (M, ρ), the broker forwards the notification to the next broker or the subscriber and outputs y_s. If S does not satisfy, the broker outputs 0.
- Retrieve. On input $(\mathsf{param}, (n_c, n_p), K_s)$ where param is the system parameter, $n_c = (C', r_2, w_1, \sigma_2, C, C_1, \ldots, C_\ell)$, $n_p = ((M, \rho), y_p, \sigma_n, (M', \rho'))$ is the notification and $K_s = (K, L, \{K_x \colon \forall x \in S\})$ is the subscriber key, suppose that S satisfies the access structure (M, ρ). The subscriber first checks the correctness of n_p as in Match and outputs $\perp$ if the checking fails. Then it finds the set $I = \{i : \rho(i) \in S\}$. Let $\{\omega \in \mathbb{Z}_p\}_{i \in I}$ be a set of constants such that if $\{\lambda_i\}$ are valid shares of any secret s according to M, then $\sum_{i \in I} \omega_i \lambda_i = s$. Then he computes

$$W = \frac{\hat{e}(C', K)}{(\prod_{i \in I}(\hat{e}(C_i, L)\hat{e}(C', K_{\rho(i)}))^{\omega_i})}.$$

[9](σ_1, σ_2) can be viewed as the sanitizable signature part of the notification. Even without the knowledge of m, the broker can check the validity of σ_2 in the Match algorithm to ensure that the notification is authenticated.

The subscriber calculates $(m, \sigma_1, r_1) = \mathsf{SKE.Dec}(1^\lambda, W, C)$. Then he computes

$$w_1 = H_2(m \parallel r_1), \qquad w_2 = H_2(M \parallel \rho \parallel r_2),$$
$$w_3 = H_2\big(w_1 \parallel w_2 \parallel C' \parallel C \parallel C_1 \parallel \cdots \parallel C_\ell\big).$$

If $\hat{e}(\sigma_1\sigma_2, g) \neq \hat{e}(w_1 w_2 w_3, y_p)$, the subscriber outputs $\perp$. Otherwise, he outputs a pair (m, y_p).

Correctness. The matching correctness is straightforward. The retrieval correctness is shown as follows:

$$\begin{aligned}
&\frac{\hat{e}(C', K)}{(\prod_{i\in I}(\hat{e}(C_i, L)\hat{e}(C', K_{\rho(i)}))^{\omega_i})} \\
&\quad= \frac{\hat{e}(g^s, g^\alpha g_1^t)}{(\prod_{i\in I}(\hat{e}(g_1^{\lambda_i} H_1(\rho(i))^{-s}, g^t)\hat{e}(g^s, H_1(\rho(i))^t)^{\omega_i})} \\
&\quad= \frac{\hat{e}(g,g)^{\alpha s}\hat{e}(g, g_1)^{st}}{(\prod_{i\in I}\hat{e}(g, g_1)^{t\lambda_i\omega_i})} = \hat{e}(g,g)^{\alpha s}.
\end{aligned}$$

5. Security analysis of our basic scheme

Our basic scheme has publisher confidentiality, information confidentiality, information unforgeability, subscription unforgeability and service unforgeability.

Theorem 1. *Suppose the (ε, t')-decisional q-BDHE assumption holds. Then our basic scheme is $(\varepsilon, t, q_s, q_p)$-selectively publisher confidential against chosen plaintext attack in the random oracle model, with a challenge filter (M^*, ρ^*) and*

$$t' = t + (q_s + q_h)O\big(n^*(\tau_m + \tau_e)\big),$$

where M^* is of size $\ell^* \times n^*$ and $n^* \leqslant q$, q_h is the number of query to the H_1 oracle, τ_m and τ_e are the time for a multiplication and an exponentiation in $\mathbb{G}$, respectively.

Proof. Assume there is a (ε, t, q_s)-adversary $\mathcal{A}$. We are going to construct another probabilistic polynomial time (PPT) $\mathcal{B}$ that makes use of $\mathcal{A}$ to solve the decisional q-BDHE problem with probability at least ε and in time at most t'. We implicitly set the manager's secret key as $g^{a^{q+1}}$ (and multiplied by a known value in $\mathbb{G}$). The value g^s is used as the randomness term in the challenge ciphertext. Therefore in the challenge ciphertext, the computation of the term C should use $\hat{e}(g,g)^{sa^{q+1}}$, which is the solution to the BDHE problem.

$\mathcal{B}$ is given the BDHE challenge $(g, g^s, g^a, \ldots, g^{a^q}, g^{a^{q+2}}, \ldots, g^{a^{2q}}, T)$. In order to use $\mathcal{A}$ to solve for the problem, $\mathcal{B}$ needs to simulate a challenger and the oracles for $\mathcal{A}$. $\mathcal{B}$ does it in the following way.

Setup. The adversary $\mathcal{A}$ gives the challenge filter (M^*, ρ^*), where M^* has $n^* \leqslant q$ columns. $\mathcal{B}$ chooses random $\alpha' \in \mathbb{Z}_p$ and implicitly sets $\alpha = \alpha' + a^{q+1}$ by letting $\hat{e}(g,g)^\alpha = \hat{e}(g^a, g^{a^q})\hat{e}(g,g)^{\alpha'}$. $\mathcal{B}$ also sets $g_1 = g^a$ and runs $(\mathsf{IBS.mpk}, \mathsf{IBS.msk}) \leftarrow \mathsf{IBS.Setup}(1^\lambda)$.

Oracle simulation. By the construction of our scheme, the IssueP oracle can be simulated by using $\mathsf{IBS.msk}$ only. The $H_2, H_3, \hat{H}_1$ and $\hat{H}_2$ oracle is simulated as normal hash function. The other oracles are simulated as follows.

- H_1 *Oracle*. On input x, if $H_1(x)$ was already defined in the table, then return the same answer as before; otherwise, choose a random value $z_x \in \mathbb{Z}_p$. If there exists an i such that $\rho^*(i) = x$, then let

  $$H_1(x) = g^{z_x} \cdot g^{aM^*_{i,1}} \cdot g^{a^2 M^*_{i,2}} \cdots g^{a^{n^*} M^*_{i,n^*}}.$$

  Otherwise, let $H_1(x) = g^{z_x}$.
  Notice that the oracle outputs are randomly distributed due to the g^{z_x} factor. Since we restrict that ρ^* is an injective function, for any x there is at most one i such that $\rho^*(i) = x$.
- *IssueS Oracle*. On input the subscription S where S does not satisfy M^*, $\mathcal{B}$ first chooses a random $r \in \mathbb{Z}_p$. Then it finds a vector $\vec{w} = (w_1, \ldots, w_{n^*}) \in \mathbb{Z}_p^{n^*}$ such that $w_1 = -1$ and for all i where $\rho^*(i) \in S$ we have $\vec{w} \cdot M^*_i = 0$.
  $\mathcal{B}$ begins by implicitly defining $t = r + w_1 a^q + w_2 a^{q-1} + \cdots + w_{n^*} a^{q-n^*+1}$ by setting

  $$L = g^r \prod_{i=1}^{n^*} \bigl(g^{a^{q+1-i}}\bigr)^{w_i}, \qquad K = g^{\alpha'} g^{ar} \prod_{i=2}^{n^*} \bigl(g^{a^{q+2-i}}\bigr)^{w_i}.$$

  Next $\mathcal{B}$ have to calculate K_x for all $x \in S$. First, if there is no i such that $\rho^*(i) = x$, $\mathcal{B}$ can simply let $K_x = L^{z_x}$. Otherwise, $\mathcal{B}$ calculates

  $$K_x = L^{z_x} \prod_{j=1}^{n^*} \left(g^r \prod_{\substack{k=1 \\ k \neq j}}^{n^*} \bigl(g^{a^{q+1+j-k}}\bigr)^{w_k} \right)^{M_{i,j}}.$$

  Notice that the term $g^{a^{q+1}}$ term cancels when combined since $M_i \cdot \vec{w} = 0$.

Challenge. The adversary gives two messages m_0^*, m_1^* and a publisher secret key x_p^* to $\mathcal{B}$. $\mathcal{B}$ flips a coin β. It chooses random $r_1, r_2, y_2', \ldots, y_n' \in \mathbb{Z}_p$ and creates

$$C' = g^s, \qquad w_1 = H_2\bigl(m_\beta^* \parallel r_1\bigr), \qquad w_2 = H_2\bigl(M^* \parallel \rho^* \parallel r_2\bigr),$$

$$\sigma_1 = w_1^{x_p}, \qquad C = \mathsf{SKE.Enc}\big(1^\lambda, H_3(T), \big(m_\beta^*, \sigma_1, r_1\big)\big).$$

$\mathcal{B}$ implicitly defines $\vec{v} = (s, sa + y_2', sa^2 + y_3', \ldots, sa^{n^*-1} + y_{n^*}')$ by setting

$$C_i = \left(\prod_{j=2}^{n^*} (g^a)^{M_{i,j}^* y_j'}\right) (g^s)^{-z_{\rho^*(i)}}.$$

$\mathcal{B}$ computes $w_3 = H_2(w_1 \parallel w_2 \parallel C' \parallel C \parallel C_1 \parallel \cdots \parallel C_\ell)$ and $\sigma_2 = (w_2 w_3)^{x_p}$. It sets $n_c^* = (C', r_2, w_1, \sigma_2, C, C_1, \ldots, C_\ell, M^*, \rho^*)$. The notification policy n_p^* can be computed using $\mathsf{IBS.msk}$ without involving m_β^*. The challenge notification is published as (n_c^*, n_p^*).

Output calculation. $\mathcal{A}$ will eventually output a guess β'. If $\beta = \beta'$, $\mathcal{B}$ then outputs his guess that $T = \hat{e}(g,g)^{a^{q+1}s}$; otherwise, $\mathcal{B}$ outputs his guess that T is a random group element in $\mathbb{G}$.

Probability analysis. Notice that only C and w_1 contain the information about m_β^*. Since H_2 is a collision resistant hash function and r_1 is also encrypted in C, m_β^* is completely hidden from $\mathcal{A}$ when T is a random group element. Therefore $\mathcal{B}$'s probability of solving the problem is the same as $\mathcal{A}$'s advantage in this game.

Time analysis. In the proof, $\mathcal{A}$ has to compute at most $\mathrm{O}(n^*)$ multiplication and exponentiation for every IssueS oracle query and H_1 oracle query. □

As discussed in Section 2.1, publisher confidentiality implies information confidentiality. Therefore our basic scheme is also selectively secure for information confidentiality against the chosen plaintext attack. However, we can give a direct proof without selective model and use a weaker assumption.

Theorem 2. *Suppose the (ε, t')-DBDH assumption holds. Then our basic scheme is (ε, t, q_p)-information confidential against chosen plaintext attack in the random oracle model, with $t' = t + q_h\mathrm{O}(\tau_e)$, where q_h is the number of query to the H_1 oracle and τ_e is the time for an exponentiation in $\mathbb{G}$.*

Proof. Assume there is a (ε, t, q_p)-adversary $\mathcal{A}$. We are going to construct another PPT $\mathcal{B}$ that makes use of $\mathcal{A}$ to solve the DBDH problem with probability at least ε and in time at most t'. We implicitly set the manager's secret key as g^{ab}. The value g^s is used as the randomness term in the challenge ciphertext. Therefore in the challenge ciphertext, the computation of the term C should use $\hat{e}(g,g)^{sab}$, which is the solution to the DBDH problem.

$\mathcal{B}$ is given the DBDH challenge (g, g^s, g^a, g^b, T). In order to use $\mathcal{A}$ to solve for the problem, $\mathcal{B}$ needs to simulate a challenger and the oracles for $\mathcal{A}$. $\mathcal{B}$ does it in the following way.

Setup. $\mathcal{B}$ implicitly sets $\alpha = ab$ by letting $\hat{e}(g,g)^\alpha = \hat{e}(g^a, g^b)$. $\mathcal{B}$ picks a random $\mu \in \mathbb{Z}_p$ and sets $g_1 = g^\mu$. The rest of the system parameters are honestly generated.

Oracle simulation. By the construction of our scheme, the IssueP oracle is not needed. The $H_2, H_3, \hat{H}_1$ and $\hat{H}_2$ oracles are simulated as normal hash function. The H_1 Oracle is simulated as follows: On input x, if $H_1(x)$ was already defined in the table, then simply return the same answer as before; otherwise, choose a random value $z_x \in \mathbb{Z}_p$ and return $H_1(x) = g^{z_x}$.

Challenge. The adversary gives two messages m_0^*, m_1^*, a challenge filter (M^*, ρ^*) and a publisher secret key x_p^* to $\mathcal{B}$. $\mathcal{B}$ flips a coin β. $\mathcal{B}$ chooses random $r_1, r_2, y_2, \ldots, y_n \in \mathbb{Z}_p$ and creates

$$C' = g^s, \qquad w_1 = H_2\big(m_\beta^* \parallel r_1\big), \qquad w_2 = H_2\big(M^* \parallel \rho^* \parallel r_2\big),$$
$$\sigma_1 = w_1^{x_p}, \qquad C = \mathsf{SKE.Enc}\big(1^\lambda, H_3(T), \big(m_\beta^*, \sigma_1, r_1\big)\big).$$

$\mathcal{B}$ implicitly defines $\vec{v} = (s, y_2, y_3, \ldots, y_{n^*})$ by setting

$$C_i = \left(\prod_{j=2}^{n^*} g_1^{M_{i,j}^* y_j}\right) \big(g^s\big)^{\mu(M_{i,1}^* - z_{\rho^*(i)})}.$$

$\mathcal{B}$ computes $w_3 = H_2(w_1 \parallel w_2 \parallel C' \parallel C \parallel C_1 \parallel \cdots \parallel C_\ell)$ and $\sigma_2 = (w_2 w_3)^{x_p}$. It sets $n_c^* = (C', r_2, w_1, \sigma_2, C, C_1, \ldots, C_\ell, M^*, \rho^*)$. The notification policy n_p^* can be computed using $\mathsf{IBS.msk}$ without involving m_β^*. The challenge notification is published as (n_c^*, n_p^*).

Output calculation. $\mathcal{A}$ will eventually output a guess β'. If $\beta = \beta'$, $\mathcal{B}$ then outputs his guess that $T = \hat{e}(g, g)^{abs}$; otherwise, $\mathcal{B}$ outputs his guess that T is a random group element in $\mathbb{G}$.

Probability analysis. Notice that only C and w_1 contain the information about m_β^*. Since H_2 is a collision resistant hash function and r_1 is also encrypted in C, m_β^* is completely hidden from $\mathcal{A}$ when T is a random group element. Therefore $\mathcal{B}$'s probability of solving the problem is the same as $\mathcal{A}$'s advantage in this game.

Time analysis. $\mathcal{A}$ has to compute at most O(1) exponentiation for every H_1 oracle query. □

Theorem 3. *Suppose the (ε', t')-CDH assumption holds. Then our basic scheme is (ε, t, q_p)-information unforgeable against chosen message attack in the random oracle model, where*

$$\varepsilon' \geqslant \varepsilon\left(\frac{3}{q_h} - \frac{3}{q_h^2} + \frac{1}{q_h^3}\right), \qquad t' = t + (q_p + q_h)\mathrm{O}(\tau_m + \tau_e),$$

where τ_m and τ_e are the time for a multiplication and an exponentiation in $\mathbb{G}$, respectively; and q_h is the number of query to the H_2 oracle.

Proof. Assume there is a (ε, t, q_p)-adversary $\mathcal{A}$. We are going to construct another PPT $\mathcal{B}$ that makes use of $\mathcal{A}$ to solve the CDH problem with probability at least ε' and in time at most t'. We set the publisher public key as g^a. We put g^b into one of the H_2 oracle's output. Therefore with non-negligible probability, $\mathcal{B}$ can extract the answer of the CDH problem g^{ab} from $\mathcal{A}$'s output signature.

$\mathcal{B}$ is given the CDH challenge (g, g^a, g^b). In order to use $\mathcal{A}$ to solve for the problem, $\mathcal{B}$ needs to simulate the oracles for $\mathcal{A}$ in the following way.

Setup. $\mathcal{B}$ chooses random $\alpha \in \mathbb{Z}_p$ and runs $(\mathsf{IBS.mpk}, \mathsf{IBS.msk}) \leftarrow \mathsf{IBS.Setup}(1^\lambda)$. It gives the master secret key $(g^\alpha, \mathsf{IBS.msk} := \beta)$ to the adversary $\mathcal{A}$. $\mathcal{B}$ also gives the publisher public key g^a to $\mathcal{A}$. The rest of the system parameters are honestly generated.

Oracle simulation. The $H_1, H_3, \hat{H}_1$ and $\hat{H}_2$ oracles are simulated as normal hash function. The other oracles are simulated as follows.

- H_2 *Oracle*. On input x, if $H_2(x)$ was already defined in the table, then return the same answer as before; otherwise, choose a random value $b_x \in \mathbb{Z}_p$. With probability $1/q_h$, the oracle outputs

$$H_2(x) = g^b \cdot g^{b_x},$$

  and stores $(x, b_x, 1)$ in the table. Otherwise, the oracle outputs $H_2(x) = g^{b_x}$ and stores $(x, b_x, 0)$ in the table. Notice that the oracle outputs are randomly distributed due to the g^{b_x} factor.
- *Pub Oracle*. On input the message m and filter (M, ρ), $\mathcal{B}$ first chooses random $r_1, r_2 \in \mathbb{Z}_p$. $\mathcal{B}$ queries the H_2 oracle and if $(m \parallel r_1, \cdot, 1)$ or $(M \parallel \rho \parallel r_2, \cdot, 1)$ appears in the table, $\mathcal{B}$ picks another random number and starts again. Otherwise, let $w_1 = g^{b_1}$ and $w_2 = g^{b_2}$. For $i = 1, \ldots, \ell$, he calculates $\lambda_i = \vec{v} \cdot M_i$, where M_i is the vector corresponding to the ith row of M. $\mathcal{B}$ chooses a random $s \in \mathbb{Z}_p$ and calculates

$$\sigma_1 = (g^a)^{b_1}, \qquad C' = g^s,$$
$$C = \mathsf{SKE.Enc}\big(1^\lambda, \hat{e}(g,g)^{\alpha s}, (m, \sigma_1, r_1)\big),$$
$$C_1 = g_1^{\lambda_1} H_1\big(\rho(1)\big)^{-s}, \ldots, \qquad C_\ell = g_1^{\lambda_\ell} H_1\big(\rho(\ell)\big)^{-s}.$$

  $\mathcal{B}$ queries the H_2 oracle and if $(w_1 \parallel w_2 \parallel C' \parallel C \parallel C_1 \parallel \cdots \parallel C_\ell, \cdot, 1)$ appears in the table, $\mathcal{B}$ picks another random number s and starts again. Otherwise, let $w_3 = g^{b_3}$. Then $\mathcal{B}$ calculates

$$\sigma_2 = (g^a)^{b_2 + b_3}.$$

  The oracle outputs the notification $(n_c^* = (C', r_2, w_1, \sigma_2, C, C_1, \ldots, C_\ell, M, \rho)$, $n_p^*)$, where n_p^* can be computed using $\mathsf{IBS.msk}$.

Output calculation. $\mathcal{A}$ will eventually output a notification $n^* = (n_c^*, n_p^*)$, where $n_c^* = (C'^*, r_2^*, w_1^*, \sigma_2^*, C^*, C_1^*, \dots, C_\ell^*, M^*, \rho^*)$. We require that n^* is not the output from the Pub oracle. $\mathcal{B}$ computes

$$\begin{aligned}
&(m^*, \sigma_1^*, r_1^*) = \mathsf{SKE.Dec}\big(1^\lambda, \hat{e}(g^\alpha, C'^*), C^*\big),\\
&w_2^* = H_2\big(M^* \parallel \rho^* \parallel r_2^*\big),\\
&w_3^* = H_2\big(w_1^* \parallel w_2^* \parallel C'^* \parallel C^* \parallel C_1^* \parallel \cdots \parallel C_\ell^*\big).
\end{aligned}$$

If the tuple $(\sigma_1^*, \sigma_2^*, w_1^*, w_2^*, w_3^*, m^*, C'^*, r_1^*, r_2^*)$ is not computed in the Pub oracle, then $\mathcal{B}$ searches the H_2 table for w_1^*, w_2^* and w_3^*. If there is no entry that $(w_i^*, b_i, 1)$ appears in the table for $i = 1, 2, 3$, $\mathcal{B}$ declares failure and exits. Without loss of generality, assume $w_1^* = g^b \cdot g^{b_1}$, $w_2^* = g^{b_2}$ and $w_3^* = g^{b_3}$. If $\mathcal{A}$ outputs a valid publication, then it satisfies:

$$\hat{e}\big(\sigma_1^*\sigma_2^*, g\big) = \hat{e}\big(w_1^* w_2^* w_3^*, g^a\big) = \hat{e}\big(g^b \cdot g^{b_1+b_2+b_3}, g^a\big).$$

Then $\mathcal{B}$ can output $\sigma_1^*\sigma_2^*/(g^a)^{b_1+b_2+b_3}$ as the solution to the CDH problem.

If the tuple $(\sigma_1^*, \sigma_2^*, w_1^*, w_2^*, w_3^*, m^*, C'^*, r_1^*, r_2^*)$ is computed in the Pub oracle, but $(C'^*, C^*, C_1^*, \dots, C_\ell^*, M^*, \rho^*)$ are different from the oracle output, it happens with negligible probability, since H_2 is collision resistant and

$$\begin{aligned}
&w_2^* = H_2\big(M^* \parallel \rho^* \parallel r_2^*\big),\\
&w_3^* = H_2\big(w_1^* \parallel w_2^* \parallel C'^* \parallel C^* \parallel C_1^* \parallel \cdots \parallel C_\ell^*\big).
\end{aligned}$$

Finally if n_c^* is the same as the corresponding value in the Pub oracle output, but the n_p^* value is different. Then it means that the new identity-based signature $\sigma_n^* = (S_1^*, S_2^*)$ has the value $S_1^* = C'$ as in the oracle. The value $C_1' = g^s$ was chosen by $\mathcal{B}$ during the Pub oracle query phase, and the value of s is unknown to $\mathcal{A}$. However, by the verification of the IBS, we have

$$S_2^* = \hat{H}_1(\mathsf{id})^{\beta \hat{H}_2(\mathsf{id}, \hat{M})} h^s.$$

If the CDH problem is hard, $\mathcal{A}$ cannot compute h^s and hence cannot compute a valid σ_n^*. We may also construct a similar security reduction to the CDH problem in this case.

Probability analysis. $\mathcal{B}$ aborts only when there is no entry that $(w_i^*, b_i, 1)$ appears in the table for $i = 1, 2, 3$. For each time, it happens with probability $1 - 1/q_h$. Therefore the probability that $\mathcal{B}$ does not abort is greater than $\frac{3}{q_h} - \frac{3}{q_h^2} + \frac{1}{q_h^3}$.

Time analysis. In the proof, $\mathcal{A}$ has to compute at most O(1) multiplication and exponentiation for every Pub oracle query and H_1 oracle query. □

Theorem 4. *Suppose that* (Sig, Vfy) *is EUF-CMA secure. Then no poly-time adversary can break the subscription unforgeability.*

Theorem 4 is straightforward by the construction of our basic scheme.

Theorem 5. *Suppose the* (ε', t')*-CDH assumption holds. Then our basic scheme is* (ε, t, q_p)*-service unforgeable against chosen message attack in the random oracle model, where*

$$\varepsilon' \geqslant \varepsilon\left(\frac{2}{q_h} - \frac{1}{q_h^2}\right), \qquad t' = t + (q_p + q_h)\mathrm{O}(\tau_m + \tau_e),$$

where τ_m *and* τ_e *are the time for a multiplication and an exponentiation in* $\mathbb{G}$, *respectively; and* q_h *is the number of query to the* H_2 *oracle.*

Proof. The setup and the oracle simulations are the same as information unforgeability. Therefore they are omitted for simplicity.

Output calculation. $\mathcal{A}$ will eventually output a notification $n^* = (n_c^*, n_p^*)$, where $n_c^* = (C'^*, r_2^*, w_1^*, \sigma_2^*, C^*, C_1^*, \ldots, C_\ell^*, M^*, \rho^*)$. We require that n^* is not the output from the Pub oracle. $\mathcal{B}$ computes

$$\begin{aligned}
&(m^*, \sigma_1^*, r_1^*) = \mathsf{SKE.Dec}\big(1^\lambda, \hat{e}(g^\alpha, C'^*), C^*\big),\\
&w_2^* = H_2\big(M^* \parallel \rho^* \parallel r_2^*\big),\\
&w_3^* = H_2\big(w_1^* \parallel w_2^* \parallel C'^* \parallel C^* \parallel C_1^* \parallel \cdots \parallel C_\ell^*\big).
\end{aligned}$$

If the tuple $(\sigma_1^*, \sigma_2^*, w_1^*, w_2^*, w_3^*, m^*, C'^*, r_1^*, r_2^*)$ is not computed in the Pub oracle, then $\mathcal{B}$ searches the H_2 table for $w_1^*.w_2^*$ and w_3^*. If there is no entry that $(w_i^*, b_i, 1)$ appears in the table for $i = 2, 3$, $\mathcal{B}$ declares failure and exits. Without loss of generality, assume $w_2^* = g^b \cdot g^{b_2}$ and $w_3^* = g^{b_3}$. If $\mathcal{A}$ outputs a valid publication, then it satisfies:

$$\hat{e}(\sigma_2^*, g) = \hat{e}(w_2^* w_3^*, g^a) = \hat{e}(g^b \cdot g^{b_2+b_3}, g^a).$$

Then $\mathcal{B}$ can output $\sigma_2^*/(g^a)^{b_2+b_3}$ as the solution to the CDH problem.

If the tuple $(\sigma_1^*, \sigma_2^*, w_1^*, w_2^*, w_3^*, m^*, C'^*, r_1^*, r_2^*)$ is computed in the Pub oracle, but $(C'^*, C^*, C_1^*, \ldots, C_\ell^*, M^*, \rho^*)$ are different from the oracle output, it happens with negligible probability, since H_2 is collision resistant and

$$\begin{aligned}
&w_2^* = H_2\big(M^* \parallel \rho^* \parallel r_2^*\big),\\
&w_3^* = H_2\big(w_1^* \parallel w_2^* \parallel C'^* \parallel C^* \parallel C_1^* \parallel \cdots \parallel C_\ell^*\big).
\end{aligned}$$

Finally if n_c^* is the same as the corresponding value in the Pub oracle output, but the n_p^* value is different. Then it means that the new identity-based signature $\sigma_n^* = (S_1^*, S_2^*)$ has the value $S_1^* = C'$ as in the oracle. The value $C_1' = g^s$ was chosen by $\mathcal{B}$ during the Pub oracle query phase, and the value of s is unknown to $\mathcal{A}$. However, by the verification of the IBS, we have

$$S_2^* = \hat{H}_1(\mathsf{id})^{\beta \hat{H}_2(\mathsf{id}, \hat{M})} h^s.$$

If the CDH problem is hard, $\mathcal{A}$ cannot compute h^s and hence cannot compute a valid σ_n^*. We may also construct a similar security reduction to the CDH problem in this case.

Probability analysis. $\mathcal{B}$ aborts only when there is no entry that $(w_i^*, b_i, 1)$ appears in the table for $i = 1, 2, 3$. For each time, it happens with probability $1 - 1/q_h$. Therefore the probability that $\mathcal{B}$ does not abort is greater than $\frac{2}{q_h} - \frac{1}{q_h^2}$.

Time analysis. In the proof, $\mathcal{A}$ has to compute at most O(1) multiplication and exponentiation for every Pub oracle query and H_1 oracle query. □

Theorem 6. *Suppose the (ε', t')-CDH assumption holds. Then our basic scheme is $(\varepsilon, t, q_I, q_p)$-publisher authenticity against chosen message attack in the random oracle model, where*

$$\varepsilon' \geqslant \varepsilon / q_{h_1'}, \qquad t' = t + q_p \mathrm{O}(\tau_m) + (q_I + q_p + q_{h_1} + q_{h_1'})\mathrm{O}(\tau_e),$$

where τ_m and τ_e are the time for a multiplication and an exponentiation in $\mathbb{G}$, respectively; and $q_{h_1}, q_{h_1'}$ are the number of query to the H_1, $\hat{H}_1$ oracle respectively.

Proof. Assume there is a $(\varepsilon, t, q_I, q_p)$-adversary $\mathcal{A}$. We are going to construct another PPT $\mathcal{B}$ that makes use of $\mathcal{A}$ to solve the CDH problem with probability at least ε' and in time at most t'. We set the $\mathsf{IBS.mpk}$ in the system parameter as g^a. We put g^b into one of the $\hat{H}_1$ oracle's output. Therefore with non-negligible probability, $\mathcal{B}$ can extract the answer of the CDH problem g^{ab} from $\mathcal{A}$'s output notification.

$\mathcal{B}$ is given the CDH challenge (g, g^a, g^b). In order to use $\mathcal{A}$ to solve for the problem, $\mathcal{B}$ needs to simulate the oracles for $\mathcal{A}$ in the following way.

Setup. $\mathcal{B}$ chooses random $\alpha \in \mathbb{Z}_p$ as part of the master secret key. $\mathcal{B}$ sets $\mathsf{IBS.mpk}$ in the system parameter as $g^\beta = g^b$ and $h = g^a$, which means $\mathsf{IBS.msk} = \beta$ is unknown. The rest of the system parameters are honestly generated. $\mathcal{B}$ runs $(x_p^*, y_p^*) \leftarrow \mathsf{KeyGen}_p(\mathsf{param})$ and also sends them to $\mathcal{A}$.

Oracle simulation. The H_2, H_3 and $\hat{H}_2$ oracles are simulated as normal hash function. The other oracles are simulated as follows.

- $\hat{H}_1$ *Oracle*. On input $x = (y_p, M, \rho)$, if $\hat{H}_1(x)$ was already defined in the table, then return the same answer as before; otherwise, choose a random value

$b_x \in \mathbb{Z}_p$. If $y_p = y_p^*$ and with probability $1/q_h$, the oracle outputs

$$\hat{H}_1(x) = g^{ab_x},$$

and stores $(x, b_x, 1)$ in the table. Otherwise, the oracle outputs $\hat{H}_1(x) = g^{b_x}$ and stores $(x, b_x, 0)$ in the table. Notice that the oracle outputs are randomly distributed due to the g^{b_x} factor.

- H_1 *Oracle*. On input x, if $H_1(x)$ was already defined in the table, then return the same answer as before; otherwise, choose a random value $a_x \in \mathbb{Z}_p$. The oracle outputs $H_1(x) = g^{a_x}$ and stores (x, a_x) in the table.
- *IssueP Oracle*. On input the $\mathsf{filter} = (M, \rho)$ and the publisher public key y_p, it asks the $\hat{H}_1$ oracle to obtain $\hat{H}_1((y_p, M, \rho))$. If it is equal to g^{ab_x}, then $\mathcal{B}$ declares failure and exits. Otherwise, it calculates the identity-based secret key $\mathsf{sk}_m = (g^b)^{b_x}$ and returns $K_p = (\mathsf{sk}_m, (M, \rho))$.
- *Pub Oracle*. On input the message m, the access structure $\mathbb{A} = (M, \rho)$ and $\mathsf{filter} = (M', \rho')$, if $H_1(y_p^*, M', \rho') = g^{b_x}$, then $\mathcal{B}$ calculates the identity-based secret key sk_m as the IssueP Oracle and computes the notification using sk_m and x_p^*.

  Otherwise, $\hat{H}_1(y_p^*, M', \rho') = g^{ab_x}$. $\mathcal{B}$ first chooses random $r, h_2 \in \mathbb{Z}_p$ and denote $s = r - bh_2 b_x$. Then it calculates

  $$S_1 = g^s = g^r\left(g^b\right)^{-b_x h_2}, \quad S_2 = \left(g^{ab_x}\right)^{bh_2} h^s = \left(g^a\right)^r.$$

  In order to compute the correct n_c without knowing the value s, $\mathcal{B}$ sets $C' = S_1$. For $C_1, \ldots, C_\ell$, we use the H_1 oracle to help to compute $H_1(x)^s = g^{a_x s} = g^{ra_x}(g^b)^{-b_x h_2 a_x}$. For the value C, $\mathcal{B}$ can compute $\hat{e}(g,g)^{\alpha s} = \hat{e}(g, S_1)^\alpha$ using the knowledge of α. Therefore $\mathcal{B}$ can calculate a valid n_c using S_1. After that, $\mathcal{B}$ sets $h_2 = \hat{H}_2(\mathsf{id}, \hat{M})$ where $\mathsf{id} = (y_p, M', \rho')$ and $\hat{M} = (n_c, M, \rho)$ for the $\hat{H}_2$ oracle. If such $\hat{H}_2$ oracle value has been set before, then $\mathcal{B}$ picks another value r and repeats the above computation. Otherwise, $\mathcal{B}$ sets $n_p = ((M, \rho), y_p^*, \sigma_n, (M', \rho'))$ and returns $n = (n_c, n_p)$.

Output calculation. $\mathcal{A}$ will eventually output a notification $n^* = (n_c^*, n_p^*)$, with $n_p^* = ((M^*, \rho^*), y_p^*, \sigma_n^*, (M', \rho'))$ for a message m^* and subscription sub^*. We require that n^* is not the output from the Pub oracle and there is no IssueP oracle query with input (filter, y_p^*) such that $\mathsf{sub}^* \in \mathsf{filter}$. If $\mathcal{A}$ wins the game, it means that σ_n^* is a valid signature on $\hat{M}^* = (n_c^*, M^*, \rho^*)$ for the identity $\mathsf{id}^* = (y_p^*, M', \rho')$. If $\hat{H}_1(y_p^*, M', \rho') \neq g^{ab_x}$, $\mathcal{B}$ declares failure and exits. Otherwise, denote $\sigma_n^* = (S_1^*, S_2^*)$ and $h_2^* = \hat{H}_2(\mathsf{id}^*, \hat{M}^*)$. Observe that $\sigma_n^*, \mathsf{id}^*, \hat{M}^*$ cannot be completely the same as the Pub oracle output. If it passes verification of the IBS, it means that:

$$\hat{e}\left(g, S_2^*\right) = \hat{e}\left(g^\beta, g^{ab_x}\right)^{h_2^*} \cdot \hat{e}\left(S_1^*, h\right) = \hat{e}\left(g^b, g^{ab_x}\right)^{h_2^*} \cdot \hat{e}\left(S_1^*, g^a\right).$$

Then it outputs $g^{ab} = (S_2^* S_1^{*-a})^{1/b_x h_2^*}$ as the solution to the CDH problem.

Probability analysis. $\mathcal{B}$ wins only when it guess the filter (M', ρ') in n^* correctly, which happens with probability $1/q_h$.

Time analysis. In the proof, $\mathcal{A}$ has to compute at most O(1) multiplication and exponentiation for every Pub oracle query and O(1) exponentiation for every H_1, $\hat{H}_1$ and IssueP oracle query. □

6. Extensions to our basic scheme

Subscriber confidentiality. Our basic scheme does not provide subscriber confidentiality. The subscriber needs to send his subscription (set of attributes) in plaintext to the brokers. We propose an extension for subscriber confidentiality, if the subscriber knows the public key of the publisher in advance.

Public key Encryption with Keyword Search (PEKS) [3] is a public key encryption scheme that encrypt the message as well as the keyword. The decryptor calculates a trapdoor function using his secret key and the keyword that he is interested in. He forwards the trapdoor information to a third party. The third party can compare the trapdoor and the encrypted keyword to see if they match or not. In PEKS, the message and the keyword are encrypted by the sender. However in the pub/sub system, the message (notification) is encrypted by the publisher and the keyword sent by the subscriber should be encrypted (for subscriber confidentiality). Therefore some modifications are needed to apply the PEKS into our scheme.

Suppose we have a PEKS scheme with the following algorithms:

- $\mathsf{KeyGen}(1^\lambda)$: On input a security parameter 1^λ, it outputs a public and private key pair pk and sk.
- $\mathsf{PEKS}(pk, w)$: On input the public key pk and a keyword w, it outputs a searchable encryption S.
- $\mathsf{Trapdoor}(sk, w)$: On input the private key sk and a keyword w, it outputs a trapdoor T_w.
- $\mathsf{Test}(T_w, S)$: On input a trapdoor T_w and a searchable encryption S, it tests whether S encrypts w. It outputs 1 for "accept" or 0 for "reject".

A PEKS scheme must satisfy that

$$\mathsf{Pr}\big[\mathsf{Test}\big(\mathsf{Trapdoor}(sk, w), \mathsf{PEKS}(pk, w)\big) = 1\big] = 1,$$

where the probability is taken over the choice of $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and the coins of all the above algorithms. The security requirements of PEKS include consistency (perfect, computational or statistical) and privacy.

Assume the subscriber knows the public key of the publisher in advance. We have the following modified CBPS protocol:

- Pub. On input $(\mathsf{param}, m, x_p, K_p)$ where param is the system parameter, m is the message, x_p is the publisher's secret key and $K_p = (M, \rho)$ is the publisher key. The publisher computes $(C', r_2, w_1, \sigma_2, C, C_1, \ldots, C_\ell, M, \rho)$ as in the basic scheme. He also calculates $T_{\rho(i)} = \mathsf{Trapdoor}(x_p, \rho(i))$ for all i. The notification is published as $n = (C', r_2, w_1, \sigma_2, C, C_1, \ldots, C_\ell, M, \rho, \{T_{\rho(i)}\})$.

- Sub. On input the set of attributes S and the publisher public key y_p, it outputs $S_x = \mathsf{PEKS}(y_p, x)$ for all $x \in S$.
- Match. On input $(\mathsf{param}, n, \{S_i\})$ where param is the system parameter, $n = (C', r_2, w_1, \sigma_2, C, C_1, \ldots, C_\ell, M, \rho, \{T_{\rho(i)}\})$ is the notification from publisher y_p and S_x is the set of subscription ciphertext, the broker first computes

$$w_2 = H_2(M \parallel \rho \parallel r_2),$$
$$w_3 = H_2(w_1 \parallel w_2 \parallel C' \parallel C \parallel C_1 \parallel \cdots \parallel C_\ell).$$

  If $\hat{e}(\sigma_2, g_2) \neq \hat{e}(w_2 w_3, y_p)$, the broker outputs $\perp$. Otherwise, the broker forwards the notification to the subscriber when there exist some i such that $\mathsf{Test}(T_{\rho(i)}, S_x) = 1$ for all x.

It is straightforward that the subscriber confidentiality is provided by the privacy of the PEKS scheme. If the PEKS scheme has perfect consistency, then the brokers in our modified pub/sub scheme can always forward the notification correctly. However, the existing PEKS schemes are all secure against CPA only.

CCA security for publisher confidentiality. The ciphertext-policy attribute-based encryption (CP-ABE) from [28] allows delegation of secret keys by deleting attributes from a key. For example, if a user has a key for the attribute set {"Professor", "CS Department"}, he can delegate a key for the attribute "Professor" only. As suggested in [28], CCA-security for publication confidentiality can be realized in the standard model by using the techniques of Canetti, Halevi and Katz [6] to the delegation system.

Anonymity. Full anonymity for both publishers and subscribers can be achieved using anonymous credentials or attribute-based signatures, as discussed in Section 3.3.

Subscriber anonymity can also be achieved in a few ways. The subscriber can compute a ring signature instead of a standard signature in the Sub protocol. The security model for subscriber anonymity has to be changed to the one similar to the anonymity of ring signatures.

The subscriber may also compute a designated verifier signature instead of a standard signature in the Sub protocol, with the subscriber hosting broker as the designated verifier. The security model for subscriber anonymity has to be changed to the one similar to the non-transferability of designated verifier signatures.

Since both methods involve non-trivial changes to the existing security model, we omit the detailed construction for simplicity.

Denial-of-service attack. Denial-of-service attack is a significant risk for pub/sub system like other communication systems. The damage may be even more severe when the attacker publishes some fake notifications and they are spread by (honest) brokers to multiple subscribers. By the broadcasting property of the pub/sub system, the damage may be amplified.d by (honest) brokers to multiple subscribers. By the broadcasting property of the pub/sub system, the damage may be amplified.

Although it is very difficult to prevent the denial-of-service attack in general, we can try to minimize the damage of such attack. Besides the out-of-band solution suggested in [27], we provide a possible solution in our security model. The service unforgeability model requires the honest brokers to check the validity of the notification before forwarding it to the next brokers. Malicious publications are dropped by the broker who firstly encounters them. Since our basic scheme has service unforgeability, it has better protection against the denial-of-service attack by malicious publications.

Furthermore, malicious subscriptions are dropped by the subscriber hosting broker who firstly encounters them. This is captured by the subscription unforgeability model. Since our basic scheme has subscription unforgeability, it has better protection against the denial-of-service attack by malicious subscriptions.

Weaker assumption. Methods to remove the restriction of the injectiveness in ρ function and to use the weaker DBDH assumption can be found in [28].

7. Related works

In this section we compare our basic CBPS scheme and the extension with the existing CBPS schemes providing confidentiality. The result of the comparison can be found in Table 1.

The scheme of Li, Lu and Shi [16] and Srivatsa and Liu [24] use prefix-preserving tree structure for information and subscription confidentiality as well as efficient matching. However, if the adversary have a large number of matching notification and subscription pairs, then the adversary may obtain some information about the prefix in the notification and subscription. Therefore they are only secure if the adversary knows a few notification and subscription pairs. They are not secure in our security model.

Khurana [10] proposed a CBPS scheme with a threshold key sharing scheme such that t out of n managers are responsible to generates keys for subscribers and publishers. It reduces the trust to a single manager. However, the group of n managers must help to calculate the notification when the notification travels from a broker to another. It greatly increases the workload of the managers.

Zhao and Sturman [32] placed a complete trust to the border brokers in their CBPS scheme. Encryption is performed between border brokers. Publishers and subscribers access pub/sub system through the access control list. Information confidentiality

Table 1

Comparison of pub/sub schemes providing confidentiality

Scheme	Conf	Unf	Anon	Auth	Pre-Shared Key	Proof	Trust
Li et al. [16]	i, s	–	–	–	Yes	No	No
Khurana [10]	I	I, V	–	–	No	No	*
Zhao and Sturman [32]	I	i, s	–	–	No	No	BB.
Raiciu and Rosenblum [22]	I, S	i, s	–	–	Yes	Yes	No
Srivatsa and Liu [24]	i, p, s	–	–	–	No	No	No
Pesonen et al. [21]	I, S	*	–	–	No	No	BB.
Zhang et al. [31]	i, s	–	–	–	No	No	No
Our Basic Scheme	I, P	I, S, V	–	P	No	Yes	No
Our Extension (in Section 6)	I, P, S	I, S, V	p, s	P	No	Yes	No

Notes: For confidentiality (Conf), I stands for information confidentiality, S stands for subscription confidentiality and P stands for publisher confidentiality. For unforgeability (Unf), I stands for information unforgeability, S stands for subscription unforgeability and V stands for service unforgeability. For anonymity (Anon), S stands for subscriber anonymity and P stands for publisher anonymity. For authenticity (Auth), P stands for publisher authenticity. A small letter means that it is secure in a weaker security model in the original cited paper only. BB. stands for border brokers. For * in the table, it will be explained in Section 7.

and authenticity is protected by this access control. The scheme is not secure in our unforgeability model.

Pesonen, Eyers and Bacon [21] also placed a trust to the border brokers in their CBPS scheme. Publishers and subscribers access pub/sub system through the access control list. Information and subscription confidentiality are protected by this access control. Since authenticated encryption is used, integrity is also protected. However, the scheme is not secure in our unforgeability model.

Raiciu and Rosenblum [22] proposed the first CBPS scheme with proof of information and subscription confidentiality. It comes with the cost of publishers and subscribers having a pre-shared key. The unforgeability of the scheme is also protected by this pre-shared key, since encryption cannot be performed without the symmetric key. The scheme is not secure in our unforgeability model.

Zhang et al. [31] proposed a CBPS scheme using a new mechanism called information foiling. The publishers and subscribers generate a set of fake messages to hide the authentic message. Their new algorithm does not fit into our model since their confidentiality is in a probabilistic sense.

8. Conclusion

In this paper, we introduced the *first* security model for different security requirements of CBPS. We proposed a new CBPS scheme that fulfills most of the security requirements concurrently. We proved its security according to our new model.

Acknowledgment

W. Susilo is supported by the ARC Future Fellowship (FT0991397).

References

[1] A. Beimel, Secure schemes for secret sharing and key distribution, PhD thesis, Department of Computer Science, Israel Institute of Technology, 1996.

[2] J. Bethencourt, A. Sahai and B. Waters, Ciphertext-policy attribute-based encryption, in: *IEEE Symposium on Security and Privacy*, IEEE Computer Society, 2007, pp. 321–334.

[3] D. Boneh, G.D. Crescenzo, R. Ostrovsky and G. Persiano, Public key encryption with keyword search, in: *EUROCRYPT 2004*, C. Cachin and J. Camenisch, eds, LNCS, Vol. 3027, Springer, 2004, pp. 506–522.

[4] S.A. Brands, *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*, MIT Press, Cambridge, MA, USA, 2000.

[5] J. Camenisch and A. Lysyanskaya, An efficient system for non-transferable anonymous credentials with optional anonymity revocation, in: *EUROCRYPT 2001*, B. Pfitzmann, ed., LNCS, Vol. 2045, Springer, 2001, pp. 93–118.

[6] R. Canetti, S. Halevi and J. Katz, A forward-secure public-key encryption scheme, in: *EUROCRYPT 2003*, E. Biham, ed., LNCS, Vol. 2656, Springer, 2003, pp. 255–271.

[7] C. Gentry and A. Silverberg, Hierarchical id-based cryptography, in: *ASIACRYPT 2002*, Y. Zheng, ed., LNCS, Vol. 2501, Springer, 2002, pp. 548–566.

[8] V. Goyal, A. Jain, O. Pandey and A. Sahai, Bounded ciphertext policy attribute based encryption, in: *ICALP (2)*, L. Aceto, I. Damgård, L.A. Goldberg, M.M. Halldórsson, A. Ingólfsdóttir and I. Walukiewicz, eds, LNCS, Vol. 5126, Springer, 2008, pp. 579–591.

[9] V. Goyal, O. Pandey, A. Sahai and B. Waters, Attribute-based encryption for fine-grained access control of encrypted data, in: *CSS 2006*, A. Juels, R.N. Wright and S.D.C. di Vimercati, eds, ACM, 2006, pp. 89–98.

[10] H. Khurana, Scalable security and accounting services for content-based publish/subscribe systems, in: *SAC 2005*, H. Haddad, L.M. Liebrock, A. Omicini and R.L. Wainwright, eds, ACM, 2005, pp. 801–807.

[11] K. Kurosawa and Y. Desmedt, A new paradigm of hybrid encryption scheme, in: *CRYPTO 2004*, M.K. Franklin, ed., LNCS, Vol. 3152, Springer, 2004, pp. 426–442.

[12] D. Lagutin, K. Visala, A. Zahemszky, T. Burbridge and G.F. Marias, Roles and security in a publish/subscribe network architecture, in: *ISCC*, IEEE, 2010, pp. 68–74.

[13] A.B. Lewko, T. Okamoto, A. Sahai, K. Takashima and B. Waters, Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption, in: *EUROCRYPT 2010*, H. Gilbert, ed., LNCS, Vol. 6110, Springer, 2010, pp. 62–91.

[14] A.B. Lewko and B. Waters, New proof methods for attribute-based encryption: Achieving full security through selective techniques, in: *CRYPTO 2012*, R. Safavi-Naini and R. Canetti, eds, LNCS, Vol. 7417, Springer, 2012, pp. 180–198.

[15] J. Li, M.H. Au, W. Susilo, D. Xie and K. Ren, Attribute-based signature and its applications, in: *ASIACCS 2010*, D. Feng, D.A. Basin and P. Liu, eds, ACM, 2010, pp. 60–69.

[16] J. Li, C. Lu and W. Shi, An efficient scheme for preserving confidentiality in content-based publish-subscribe systems, Technical Report GIT-CC-04-01, Georgia Institute of Technology, 2004.

[17] K. Miyazaki, S. Susaki, M. Iwamura, T. Matsumoto, R. Sasaki and H. Yoshiura, Digital documents sanitizing problem, IEICE Technical Report ISEC2003-20:61–67, 2003.

[18] P. Nikander and M.F. Giannis, Towards understanding pure publish/subscribe cryptographic protocols, in: *16th International Workshop on Security Protocols*, 2008.

[19] T. Okamoto and K. Takashima, Fully secure functional encryption with general relations from the decisional linear assumption, in: *CRYPTO 2010*, T. Rabin, ed., LNCS, Vol. 6223, Springer, 2010, pp. 191–208.

[20] T. Okamoto and K. Takashima, Fully secure unbounded inner-product and attribute-based encryption, in: *ASIACRYPT 2012*, X. Wang and K. Sako, eds, LNCS, Vol. 7658, Springer, 2012, pp. 349–366.

[21] L.I.W. Pesonen, D.M. Eyers and J. Bacon, Encryption-enforced access control in dynamic multi-domain publish/subscribe networks, in: *DEBS 2007*, ACM International Conference Proceeding Series, Vol. 233, ACM, 2007, pp. 104–115.

[22] C. Raiciu and D.S. Rosenblum, Enabling confidentiality in content-based publish/subscribe infrastructures, in: *Securecomm 2006*, IEEE, 2006.

[23] A. Sahai and B. Waters, Fuzzy identity-based encryption, in: *EUROCRYPT 2005*, R. Cramer, ed., LNCS, Vol. 3494, Springer, 2005, pp. 457–473.

[24] M. Srivatsa and L. Liu, Secure event dissemination in publish-subscribe networks, in: *ICDCS 2007*, IEEE Computer Society, 2007, p. 22.

[25] M. Suzuki, T. Isshiki and K. Tanaka, Sanitizable signature with secret information, in: *Symposium on Cryptography and Information Security*, 2006, 4A1–4A2.

[26] K. Visala, D. Lagutin and S. Tarkoma, Security design for an inter-domain publish/subscribe architecture, in: *Future Internet Assembly*, J. Domingue, A. Galis, A. Gavras, T. Zahariadis, D. Lambert, F. Cleary, P. Daras, S. Krco, H. Müller, M.-S. Li, H. Schaffers, V. Lotz, F. Alvarez, B. Stiller, S. Karnouskos, S. Avessta and M. Nilsson, eds, LNCS, Vol. 6656, Springer, 2011, pp. 167–176.

[27] C. Wang, A. Carzaniga, D. Evans and A.L. Wolf, Security issues and requirements for internet-scale publish-subscribe systems, in: *HICSS 2002*, IEEE Computer Society, 2002.

[28] B. Waters, Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization, in: *PKC 2011*, D. Catalano, N. Fazio, R. Gennaro and A. Nicolosi, eds, LNCS, Vol. 6571, Springer, 2011, pp. 53–70.

[29] T.H. Yuen, W. Susilo, J.K. Liu and Y. Mu, Sanitizable signatures revisited, in: *CANS 2008*, M.K. Franklin, L.C.K. Hui and D.S. Wong, eds, LNCS, Vol. 5339, Springer, 2008, pp. 80–97.

[30] T.H. Yuen, W. Susilo and Y. Mu, Towards a cryptographic treatment of publish/subscribe systems, in: *CANS 2010*, S.-H. Heng, R.N. Wright and B.-M. Goi, eds, LNCS, Vol. 6467, Springer, 2010, pp. 201–220.

[31] H. Zhang, A. Sharma, H. Chen, G. Jiang, X. Meng and K. Yoshihira, Enabling information confidentiality in publish/subscribe overlay services, in: *ICC 2008*, IEEE, 2008, pp. 5624–5628.

[32] Y. Zhao and D.C. Sturman, Dynamic access control in a content-based publish/subscribe system with delivery guarantees, in: *ICDCS 2006*, IEEE Computer Society, 2006, p. 60.