Collusion Defender: Preserving Subscribers' Privacy in Publish and Subscribe Systems*

Shujie Cui, Sana Belguith, Pramodya De Alwis, Muhammad Rizwan Asghar, and Giovanni Russello

***Abstract*—The Publish and Subscribe (pub/sub) system is an established paradigm to disseminate the data from publishers to subscribers in a loosely coupled manner using a network of dedicated brokers. However, sensitive data could be exposed to malicious entities if brokers get compromised or hacked; or even worse, if brokers themselves are curious to learn about the data. A viable mechanism to protect sensitive publications and subscriptions is to encrypt the data before it is disseminated through the brokers. State-of-the-art approaches allow brokers to perform encrypted matching without revealing publications and subscriptions. However, if malicious brokers collude with malicious subscribers or publishers, they can learn the interests of innocent subscribers, even when the interests are encrypted.**

In this article, we present a pub/sub system that ensures confidentiality of publications and subscriptions in the presence of untrusted brokers. Furthermore, our solution resists collusion attacks between untrusted brokers and malicious subscribers (or publishers). Finally, we have implemented a prototype of our solution to show its feasibility and efficiency.

***Index Terms*—Collusion Resistance, Secure Pub/sub, Subscribers' Privacy, Publications' Confidentiality**

I. Introduction

Publish and subscribe (pub/sub) systems enable dissemination of data from *publishers* to interested *subscribers* in a loosely-coupled manner, where the data is transmitted without establishing direct contacts between publishers and subscribers. Basically, *publications*, representing the data generated by publishers, are routed to interested subscribers using a network of dedicated servers, referred to as *brokers*. These brokers form a network and could easily be offered as Software as a Service (SaaS) by cloud service providers. Typically, a publication is composed of content and a set of *tags* defining keywords that characterise its content. Subscribers register their interests (*a.k.a. subscriptions*) in publications through a set of constraints on these tags. To identify whether a subscriber is interested in receiving specific publications, brokers match the publications' tags against the registered interests. Then, the broker identifies the intended subscribers and forwards the publications to them.

*This work is an extension of initial work appeared in the proceedings of the 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications (Trustcom) 2018 under the title "Malicious Entities are in Vain: Preserving Privacy in Publish and Subscribe Systems" by Shujie Cui, Sana Belguith, Pramodya De Alwis, Muhammad Rizwan Asghar, and Giovanni Russello [1].

Shujie Cui, Sana Belguith, Muhammad Rizwan Asghar, and Giovanni Russello are with the Cyber Security Foundry, The University of Auckland, New Zealand. Sana Belguith is with the School of Computing, Science and Engineering, University of Salford, Manchester, UK. They can be contacted by email: s.cui@imperial.ac.uk, S.Belguith@salford.ac.uk, r.asghar@auckland.ac.nz, and g.russello@auckland.ac.nz, respectively.

Thanks to its characteristics, the pub/sub model has been widely used in several applications. For instance, e-health information systems [2], [3] use the pub/sub model to share health records between involved parties, *i.e.,* hospitals, doctors, and pharmacies. Another example is that of stock exchange services that deploy pub/sub systems to communicate available trades to consumers [3]–[5]. Google offers *Cloud Pub/Sub*, which is a real-time messaging service for stream analytic and event-driven computing systems [6]. These are few applications among many others.

Despite its benefits, pub/sub systems present several security and privacy challenges as the data is routed through a set of brokers in a multi-party distributed system. Indeed, publishers (or subscribers) may send (or receive) sensitive publications, such as health information, religious, and political interests. Thus, the brokers could collect sensitive information about the publishers and subscribers. With the proliferation of outsourced systems, pub/sub services are typically based on third party servers (*e.g.,* cloud servers). Unfortunately, these servers can be compromised or hacked. For instance, in 2016, an attack on the Yahoo platform led to the leakage of 1 billion user accounts [7]. Since brokers handle sensitive data and could be compromised, it is reasonable to treat them as untrusted entities and ensure the protection of publications and subscriptions.

To protect sensitive information from untrusted brokers, several works propose to encrypt the publications and subscriptions in such a way that the brokers can still match the subscriptions against the publications' tags without learning their content [8]–[12]. As a result, subscriptions and publications are protected from brokers. However, it is still possible for malicious brokers to collude with subscribers and publishers. Specifically, as described in [13], a malicious subscriber could collude with a broker by disclosing the content of her subscriptions. Consequently, even if the subscription from an innocent subscriber is encrypted, the broker can still infer the content by checking if the subscriptions from both an innocent subscriber and a malicious subscriber match the same publication tags. Likewise, a malicious publisher could mount a data injection attack, *i.e.,* publish a fake publication to learn subscribers' interests. Specifically, a malicious publisher can collude with a broker to reveal the interests matching the fake publication. Therefore, to effectively ensure the privacy of subscriptions, it is also necessary to resist collusion attacks between brokers, publishers, and subscribers. The technique against colluding subscribers and brokers was first studied by Rao *et al.* in [13]. Unfortunately, there is little work done on collusion attacks in the context of secure pub/sub systems [14]. In our literature

review, we found that only the schemes proposed in [13], [15], [16] resist collusion attacks between malicious subscribers (or publishers) and brokers. However, all these approaches require publishers and subscribers to communicate directly to protect their privacy against colluding parties. As a result, the loosely-coupled property of the pub/sub model is no longer supported by these approaches.

In this article, we provide a privacy-preserving pub/sub system that protects subscriptions effectively and resists collusion attacks using a multi-broker setting without compromising the loosely-coupled property of the pub/sub model. The novelty of our proposal lies in the use of multiple types of brokers to match and to route publications to the intended subscribers. The main idea is to divide the match operations (between encrypted subscriptions and publication tags) into different phases, where each phase is executed by a different type of broker. Each broker type only processes partial information from which it cannot infer sensitive information about the subscriptions. Thus, if a broker is compromised or colludes with a subscriber (or a publisher), the subscriptions are still protected.

Our contributions are multi-fold. First, using a scheme like Key Policy Attribute-Based Encryption (KP-ABE), publications' content can be accessed only by the authorised subscribers. Second, we apply Searchable Encryption (SE) [17] to ensure encrypted matching of publications' keywords against subscribers' interests. Third, thanks to the use of multiple brokers, the proposed solution is secure against collusion attacks between brokers and subscribers/publishers. Herein, we stress that the idea of using multiple types of brokers to defend against collusion attacks in pub/sub systems has been proposed in our previous work [1]. This work extends our idea by giving a detailed architecture, a comprehensive security analysis, and a thorough performance evaluation. Furthermore, we give a motivating scenario, identify security requirements for pub/sub systems, and present a technical background on the applied cryptographic techniques, including KP-ABE and SE schemes.

The remainder of this article is organised as follows. Section II describes a motivating scenario and lists our security requirements. Section III surveys related work. Afterwards, we present in Section IV both the system model and the threat model together with a brief overview of our approach. Next, we give the technical background in Section V before explaining solution details in Section VI. In Section VII, we provide a security analysis of our solution and give a complexity analysis in Section VIII. In Section IX, we report the performance analysis of the implementation of our proposed system. Finally, we conclude this article in Section X highlighting some future research directions.

II. Motivating Scenario and Security Requirements

To illustrate our solution, we use a scenario based on e-health systems. In e-health systems, medical entities (such as doctors, hospitals, clinics, and pharmacists) benefit from pub/sub services by employing private or public brokers to share patients' Electronic Health Records (EHR) (*cf.* Fig. 1).

To effectively diagnose and treat patients, a publisher, say a doctor from hospital A, may need to share an EHR with other authorised entities from hospital B, including doctors, pharmacists, and medical laboratory personnel. In this case, the EHR must be routed to various health organisations, possibly geographically separated and in independent administrative domains, where the patient can be moved when her conditions stabilise or tests have to be performed or analysed. Considering the shared EHR contains personal information about the patient such as her identity, address and type of injury, it has to be protected from unauthorised accesses. Moreover, subscriptions are also highly sensitive information as they can reveal which patient is treated by which clinic, for which type of disease, and other information related to the subscribers that could reveal details of the patient's conditions. To protect the patients' privacy, the pub/sub system should not leak any sensitive information related to medical personnel and patients' EHRs.

To provide a secure privacy-preserving pub/sub service, the system should protect the confidentiality of both publications and subscriptions. Based on the aforementioned e-health scenario, we define the following security requirements:

- R1. The published data should be protected from brokers and unauthorised subscribers, *i.e.,* the publications should not be accessed by brokers and unauthorised subscribers whose interests do not match the publications' tags, even if they collude together.
- R2. The brokers should be able to check if subscribers' interests match the publication tags without knowing the content (which can reveal information about the publication payload and subscriptions).
- R3. The publishers and subscribers should be loosely-coupled. In particular, publishers and subscribers should not communicate directly. When publishing messages, the publishers should not need to know the identity/location of the subscribers that might be interested in their publications. Likewise, when receiving publications, the subscribers should not be aware of the publishers' identity/location.
- R4. If colluding with malicious publishers and/or subscribers, the brokers should not be able to learn the interests of innocent subscribers, even in the presence of injected publications and subscriptions.

III. Related Work

Table I summarises how existing pub/sub systems presented in the related literature fare in respect to our requirements. Unfortunately, none of the proposed works achieves all of the aforementioned requirements.

In [18], Raiciu *et al.* introduce a secure pub/sub system that ensures confidentiality of publications and subscriptions from brokers. By combining with different SEs based on the type of values, their system supports encrypted filtering for both equality and range interests (R2: ✔). However, in their solution, the secrets for data encryption are shared among publishers and subscribers, and the publication payload is encrypted with symmetric encryption. Sharing secrets reduces

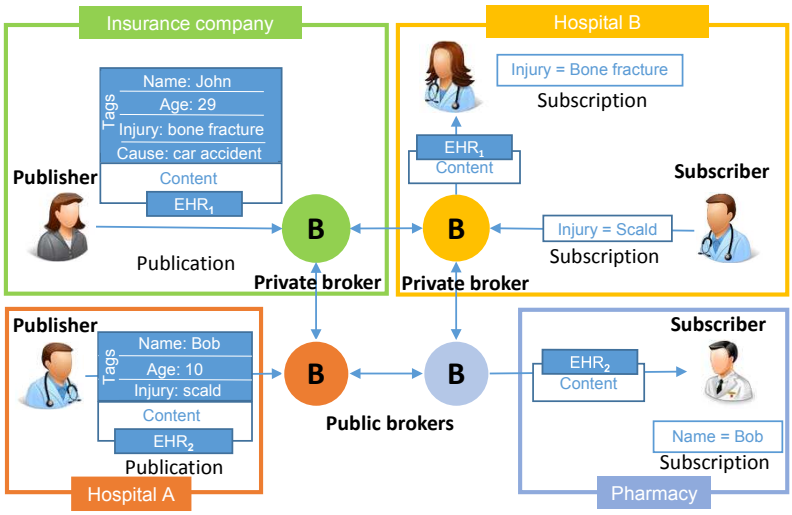


Fig. 1: E-health pub/sub Service: Several brokers are connected into a virtual cluster. Each broker belongs to a different trust domain. The publishers forward patients' Electronic Health Records (EHRs) to their local brokers, then the publications are forwarded to the intended subscribers via other brokers.

TABLE I: A comparison of pub/sub schemes.

Schemes	R1	R2	R3	R4
Raiciu *et al.* [18]	✘	✔	✘	✘
Nabeel *et al.* [19]	✘	✔	✘	✘
Nabeel *et al.* [20]	✘	✔	✔	✘
Di Crescenzo *et al.* [21]	✘	✔	✔	✘
Choi *et al.* [22]	✘	✔	✔	✘
Borcea *et al.* [23]	✔	✘	✔	✘
Tariq *et al.* [24]	✔	✔	✔	✘
Ion *et al.* [10]	✔	✔	✔	✘
Asghar *et al.* [9]	✔	✔	✔	✘
Yang *et al.* [8]	✔	✔	✔	✘
Rao *et al.* [13] [15]	✘	✔	✘	✔
Pires *et al.* [16]	✔	✔	✘	✔
Di Crescenzo *et al.* [25]	✔	✔	✔	✘
Onica *et al.* [26]	✔	✔	✔	✘
Our goals	✔	✔	✔	✔

✔and ✘indicate whether the requirement is fulfilled or not, respectively.

the decoupling of the pub/sub system (R3: ✘). If malicious subscribers/publishers reveal the shared secrets to the broker, they can learn all the publications (R1: ✘). Moreover, in their scheme, the encrypted subscription is deterministic, which leaks the relationship between interests directly. If the broker colludes with malicious subscribers, it can infer other subscribers' interests (R4: ✘).

In [19], Nabeel *et al.* present an approach based on both symmetric and asymmetric schemes. Specifically, the publication payload is encrypted with a symmetric algorithm, and both tags and subscriptions are encrypted with the Paillier homomorphic cryptosystem [27], such that the brokers can perform privacy-preserving matching over encrypted data (R2: ✔). This solution offers confidentiality of publications and subscriptions. However, it breaks the decoupling property of pub/sub system, since the subscribers have to communicate with publishers to get the subscriptions blinded (R3: ✘). This issue has been solved in [20] by using the modified Paillier cryptosystem and the Attribute-Based Group Key Management (AB-GKM) scheme [28]. However, if subscribers collude with the broker, they can still access unauthorised publications (R1: ✘). Furthermore, they cannot prevent the broker from inferring the subscriber's interests by colluding with malicious subscribers or publishers (R4: ✘).

Di Crescenzo *et al.* [21] design a 3-party pub/sub protocol that safeguards privacy of subscriptions and publications while guaranteeing the performance of the system. In the protocol, both interests and tags are encrypted with 2-layer cryptographic pseudonyms, and the encrypted tags and interests are semantically secure. A trusted third party server is employed to perform the second layer of encryption. Due to the assistance of the third party, the broker is able to test the equality between encrypted tags and interests efficiently and securely (R2: ✔). Moreover, the publishers and subscribers are not required to communicate directly (R3: ✔). However, in this protocol, the publication payload is encrypted with a key shared among all the subscribers and publishers, which will put all the publications at risk if a broker colludes with malicious subscribers and/or publishers (R1: ✘ and R4: ✘).

In [22], Choi *et al.*introduce a method to route publications to intended subscribers allowing brokers to perform matching without learning the content of the publications and subscriptions (R2: ✔), and without requiring direct communication between publishers and subscribers (R3: ✔). This proposal relies on the Asymmetric Scalar-Product Preserving Encryption (ASPE) [29] technique, which is a geometric transformation that supports sum, minimum, maximum, and count functions other than equality filtering. However, in ASPE, the subscriber knows the transformation matrix. If the broker and malicious subscribers collude, they can decode the publications and other subscriptions (R1: ✘ and R4: ✘). To mitigate this issue, the authors propose to involve a trusted third party to encrypt and forward subscriptions to the broker. However, the trusted

third party would be the bottleneck when a vast number of subscribers enrolling the system.

In [23], Borcea *et al.* propose PICADOR, a secure topic-based pub/sub system based on a proxy-re-encryption scheme. The authors apply a lattice-based proxy re-encryption scheme that allows partial homomorphic operations and ensures the loosely-coupling property of the pub/sub system (R3: ✔). That is, the brokers have to re-encrypt the publications such that only the authorised subscribers could recover the plaintext of these publications (R1: ✔). However, this re-encryption increases the computation overhead significantly on the broker end. Moreover, the topic of each publication is sent to the broker in plaintext (R2: ✘ and R4: ✘).

In [24], Tariq *et al.* present a secure broker-less pub/sub system, where the publications are distributed by honest-but-curious publishers. In their system, publications are encrypted using the Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [30], ensuring only authorised subscribers can recover the publications (R1: ✔). Moreover, they utilise Public-key Encryption with Keyword Search (PEKS) [31] to support search operations for matching encrypted tags and interests (R2: ✔). To ensure the decoupling property, the publication is encrypted with each credential that matches the tags. By doing so, the publisher does not need to know who subscribes for the publication (R3: ✔). However, this strategy increases the computation overhead on the publishers and the communication overhead significantly. Moreover, the authors do not take any measure to defend against the collusion attacks (R4: ✘).

In [10], Ion *et al.* present a pub/sub system that ensures confidentiality of publications and subscriptions. Their scheme allows the publishers to express fine-grained access control policies on the publications by applying Attribute-Based Encryption (ABE) [30] to the payload (R1: ✔). Moreover, by using the proxy-encryption [32], their scheme does not require the publishers and subscribers to share any key (R3: ✔), and the broker can check if encrypted tags match encrypted subscriptions (R2: ✔). However, this scheme does not offer any protection from collusion attacks between brokers and publishers/subscribers (R4: ✘).

PIDGIN [9] has been proposed to ensure subscriptions' privacy and publications' confidentiality in pub/sub systems. In this proposal, the publication payload is encrypted using CP-ABE with respect to access structures (R1: ✔ and R3: ✔). The publication tags and subscriptions are encrypted using PEKS, allowing the broker to perform the matching over them without requiring access to the content (R2: ✔). However, if the broker colludes with a subscriber, the broker will be able to infer the interests of innocent subscribers (R4: ✘).

Yang *et al.* [8] introduce a dual-policy ABE scheme that ensures an efficient and secure keyword search in cloud-based pub/sub systems (R2: ✔ and R3: ✔). In this proposal, the publisher defines an access policy over the publications' keywords while the subscriber sets a different access policy through its interests (R1: ✔). Basically, the publishers are considered fully trusted, the subscribers are malicious and the cloud server is curious. Moreover, they assume that the subscribers can collude together to access the publications but cannot collude with the cloud server (R4: ✘).

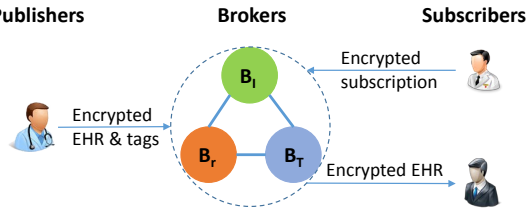


Fig. 2: An Overview of Our System: Three brokers B_I, B_r, and B_T in different domains are connected into a virtual cluster. The publishers in these domains send publications to the cluster. The three brokers in the cluster perform the matching and routing separately, and finally only the subscribers whose interests match the tags could get the publications.

Although the aforementioned solutions ensure the publications' confidentiality, they do not consider privacy of subscriptions against colluding brokers and subscribers [14]. In fact, a malicious subscriber can share her subscriptions in cleartext with the broker, which can leak the subscriptions of innocent subscribers. This issue was addressed by Rao *et al.* in [13] and [15]. Specifically, Rao *et al.* use a trusted engine to cloak and encrypt the subscriptions before sending to the broker (R4: ✔). As a result, the precise interests are hidden from brokers. However, the subscribers receive more publications than they require and have to filter out the redundant publications by performing about matching round. Moreover, the trusted engine can be a bottleneck in the system as it must remain active and uncorrupted throughout the lifetime of the system. To protect publications and subscriptions from brokers, they follow the idea given in [18], which supports encrypted filtering (R2: ✔) but needs secrets sharing among subscribers and publishers. Thus, their schemes also reduce the decoupling property (R3: ✘) and do not offer any access control mechanism over the publications (R1: ✘).

More recently, Pires *et al.* [16] present a pub/sub routing engine that leverages the trusted execution environment offered by shielded SGX enclaves [33]. In this approach, subscriptions are stored in the trusted SGX enclave and the matching operation between interests and tags is also performed by the SGX enclave (R2: ✔). In this case, if the brokers collude with malicious publishers or subscribers, they cannot infer other subscribers' subscriptions, since the brokers cannot access the results of the match operations because it is performed in the enclave (R1: ✔ and R4: ✔). However, the subscribers have to first send the subscription for re-encryption to the publishers, which violates the decoupling property of pub/sub systems (R3: ✘).

IV. Solution Overview

In this section, we begin by presenting the system model and the threat model of our solution. Then, we provide a brief overview of the approach.

A. System Model

As shown in Fig. 2, we consider a privacy-preserving pub/sub service involving the following entities:

- ***Publishers (Pub).*** The publisher generates publications and the related tags. Before publishing to the broker, it encrypts both the tags and the payload of the publication.
- ***Subscribers (Sub).*** Each subscriber defines a subscription according to its interests, such that it receives only the publications whose tags satisfy the subscription.
- ***Broker (B).*** The broker is responsible for filtering and delivering publications to the interested Subs. As shown in Fig. 2, our solution involves three types of brokers, which cooperate to perform the filtering and delivering of publications (more details are given in Section VI). Here, we stress that different types of brokers not only perform different functionalities, but also are managed by different domains.
- ***Trusted Authority (TA).*** The trusted authority is responsible for managing the keys of Subs and Pubs.

B. Threat Model

In this work, we assume the TA is fully trusted and the channels between the TA and the Pubs/Subs are secure. In our system, we consider the following threat model:

- ***Malicious Sub.*** A malicious Sub may try to access unauthorised publications and infer other Subs' interests by colluding with brokers.
- ***Malicious Pub.*** A malicious Pub may try to infer Subs' interests by injecting malicious publications and colluding with brokers.
- ***Honest but Curious Broker.*** The brokers are semi-trusted (honest-but-curious) entities. They obey the protocol to evaluate the subscriptions but they are curious about the content of publications and interests. Moreover, a broker may collude with any Sub or Pub to infer other Subs' interests. In our setting, we require three types of brokers that are managed by at least three different non-colluding domains. Moreover, we assume that the malicious Subs and Pubs could collude with at most two types of brokers managed by two different domains.

C. Our Approach

In this article, we aim at providing a pub/sub service that could protect publications and Subs' interests from curious brokers in the presence of malicious Subs and Pubs. To protect the publications from unauthorised entities, the Pub encrypts the publication using the Key-Policy Attribute-Based Encryption (KP-ABE) scheme [34]. In this way, only the authorised Subs can recover the content of the publications (R1: ✔). Note that other well-established techniques that can ensure fine-grained access control, such as [35] and [36], could also be utilised to achieve R1 in our system. Tags and interests are encrypted using an SE scheme. Thus, the brokers could check if the publication tags match Subs' interests in an encrypted manner, and distribute the publication to authorised Subs (R2: ✔). Furthermore, the TA generates the keys for KP-ABE and SE and distributes them to Subs and Pubs. This means our system does not require the Subs and Pubs to communicate directly to share the keys (R3: ✔).

Encrypting Subs' interests using SE is not sufficient to protect the subscriptions from the collusion attacks between the broker and malicious Pubs or Subs. As mentioned above, when the broker colludes with malicious Pubs, it can inject compromised publications and infer the interests of innocent Subs by observing if they match the injected publications. Likewise, when the broker colludes with malicious Subs, it can inject compromised subscriptions and infer the interests of innocent Subs by observing if they match the same publications as the injected ones. In this work, we solve this issue whereby Subs' interests are kept protected even when a broker colludes with a malicious Pub or Sub. Unlike state-of-the-art pub/sub systems that fundamentally use a single broker to match and forward the publications to the Subs, the novelty of our work lies in the use of three different types of brokers. Basically, the three types of brokers are employed with different functionalities and managed in different domains. The main idea of our solution is to divide the match operations between interests and tags into three different phases where each phase is performed by a different type of broker. In our system, we allow at most two types of brokers to collude and still be able to protect the content of the interests. Each type of broker only knows some partial information, from which sensitive information about encrypted interests cannot be inferred. Thus, even if malicious Subs or Pubs colludes with any two types of the brokers (out of the three types supported in our solution), they are unable to infer the interests of innocent Subs (R4: ✔).

Note that, our solution can be generalised to resist the collusion among $\mathcal{N}(\mathcal{N} > 1)$ brokers and malicious Subs by employing $\mathcal{N}+1$ types of brokers and dividing the match operation into $\mathcal{N}+1$ phases, where each phase is performed by one type of broker (see Section VI-D for more details). Only when malicious Pubs or Subs collude with all the $\mathcal{N}+1$ types of brokers, they could learn the subscriptions of innocent Subs. However, colluding with all types of brokers will be harder when $\mathcal{N}$ is bigger. In this work, we take $\mathcal{N}=2$ as an example to show how our solution works.

The details of our solution and the security analysis against collusion attacks are given in Section VI and VII, respectively.

V. Technical Background

In this section, we briefly cover the KP-ABE and SE schemes used in our solution.

A. KP-ABE

In KP-ABE, the ciphertext is labelled with a set of attributes while the users' private keys are associated with a non-monotonic access policy [37]. The user is able to decrypt the ciphertext if the access policy is satisfied by the attributes embedded in the ciphertext. In this work, a Pub takes the tags of each publication as the attributes sets and encrypts the publications using KP-ABE. The private keys for decrypting the publications are associated with the Subs' interests. Only the Subs whose interests satisfy the tags are able to recover the publication. KP-ABE consists of the following four algorithms:

- $\texttt{Setup}_{\texttt{KP}-\texttt{ABE}}(\lambda)$– This algorithm is performed by the TA and takes as input a security parameter λ. It generates the public parameters PP as well as a master secret key MK.
- $\texttt{Encrypt}_{\texttt{KP}-\texttt{ABE}}(\texttt{PP}, C, \tau)$– It takes a message C, the public parameters PP, and a set of attributes τ as input. It outputs the ciphertext C^*. In this work, the publication tags are regarded as the attributes set.
- $\texttt{KeyGen}_{\texttt{KP}-\texttt{ABE}}(\texttt{MK}, \Psi)$– This algorithm takes as input an access structure Ψ and the master secret key MK. It outputs a secret key sk. In this work, Sub's subscription is regarded as the access structure. This algorithm is also performed by the TA.
- $\texttt{Decrypt}_{\texttt{KP}-\texttt{ABE}}(sk, C^*)$– It takes as inputs the Sub's secret key sk for access structure Ψ and the ciphertext C^*, and outputs the message C.

B. Searchable Encryption

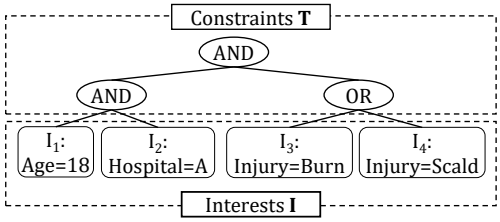
In our solution, to encrypt interests and tags we use the SUISE symmetric SE scheme [38]. On the one hand, SUISE ensures semantically secure encrypted subscriptions. Thus, the broker cannot tell if Subs have the same interests or not just based on their ciphertext. On the other hand, in SUISE, the matching between encrypted tags and interests is performed with a keyed hash function, which is much more efficient than asymmetric encryption-based schemes. Specifically, its *AddToken* and *SearchToken* primitives are used to encrypt interests and tags, respectively, and the *Search* primitive is used to match the encrypted tags with encrypted interests. Here, we emphasise that our solution is independent of any specific functionality offered by SUISE: other SE schemes that could effectively ensure confidentiality of interests and tags, such as [31], [39]–[41], could also be leveraged in our system.

The SUISE primitives involved in our system include:

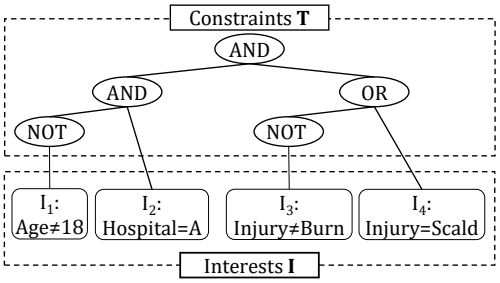
- $\texttt{Gen}_{\texttt{SE}}(\kappa)$: This algorithm is also performed by the TA. It takes as input a security parameter κ, and generates a secret key k.
- $\texttt{AddToken}_{\texttt{SE}}(\texttt{k}, \texttt{I})$: It takes as input the key k and an interest I. It generates a searchable encrypted interest I^* by computing $I^* = (H_{\mathcal{F}_k(I)}(\eta), \eta)$, where η is a nonce, $\mathcal{F} : \{0,1\}^\kappa \times \{0,1\}^* \rightarrow \{0,1\}^\kappa$ is a Pseudo-Random Function (PRF), and $H : \{0,1\}^\kappa \times \{0,1\}^* \rightarrow \{0,1\}^\kappa$ is a random oracle.
- $\texttt{SearchToken}_{\texttt{SE}}(\texttt{k}, \tau)$: Given the secret key k and a tag τ, this algorithm produces a trapdoor $\tau^* = \mathcal{F}_k(\tau)$.
- $\texttt{Search}_{\texttt{SE}}(I^*, \tau^*)$ It takes as input an encrypted interest $I^* = (\mu, \nu)$, and a trapdoor τ^* of the tag. It outputs 1 if $H_{\tau^*}(\nu) = \mu$, and 0 otherwise.

VI. Solution Details

In this section, we first describe the representations for publications and subscriptions. Second, we discuss our strategies for resisting collusion attacks. Finally, we provide the detailed description of our proposed privacy-preserving multi-broker system.



(a) The original tree structure of the subscription 'Age = 18 and Hospital=A and (Injury=Burn or Injury=Scald)'.



(b) The tree structure of the subscription 'not (Age ≠ 18) and Hospital=A and (not (Injury ≠ Burn) or Injury=Scald)'.

Fig. 3: The tree structure of a subscription: Each leaf node represents an interest I. The non-leaf nodes define the constraints over interests. Formally, each subscription is defined as: $S = (T, \mathbf{I})$, where T is the structure of the non-leaf nodes and $\mathbf{I} = (I_1, I_2, I_3, I_4)$ is the ordered set of the leaf nodes.

A. Data Representation

Formally, each publication P in our system is represented as a tuple $(ID, \boldsymbol{\tau}, C)$, where ID is the identifier, $\boldsymbol{\tau} = \{\tau_1, \ldots\}$ is a collection of *tags*, and C represents the publication's payload. To ensure the uniqueness of ID, it can be derived from C and the publisher's identifier. Each tag τ is defined as an $attribute : value$-pair, *e.g.,* 'Injury:Scald' means that the attribute 'Injury' has the value 'Scald'.

In our solution, each subscription S is represented as a tree. The leaves of the tree represent the *interests* and are defined as an $attribute = value$ pair. The non-leaf nodes of the tree denote the *constraints* over the interests, *e.g.,* 'AND' and 'OR'. Interests will match tags when they have the same $attribute$ and $value$. For instance, the interest 'Injury=Scald' will match all the tags that have the attribute 'Injury' and the value 'Scald'. Formally, $S = (\mathbf{I}, T)$, where $\mathbf{I} = (I_1, \ldots)$ represents the ordered interests set and T is the constraint between the interests. Fig. 3(a) shows the tree structure for the subscription 'Age = 18 and Hospital=A and (Injury=Burn or Injury=Scald)'. In this example, $S = (\mathbf{I} = (Age = 18, Hospital = A, Injury = Burn, Injury = Scald), T = (()and())and(()or()))$.

B. Resisting Collusion Attacks

The main goal of this work is to prevent brokers and malicious Pubs and Subs from learning the innocent Subs' subscriptions even if they collude with each other. Before giving more details of our system, we first show why previous encrypted pub/sub systems, such as [9], [18], [19], fail to

resist the collusion between brokers and malicious Pubs or Subs. Basically, in these systems, only one type of broker is used, and this type of broker is responsible for performing all the functionalities, including storing subscriptions, filtering and forwarding publications to Subs. As a consequence, the broker knows the match result between each (I, τ) pair and whether Subs have the same interests or not. When a malicious Pub exposes the plaintext of τ to the broker, they can infer the content of I based on the match result. For instance, if τ is 'Injury:Scald', the interest matching this tag must be 'Injury=Scald'. Likewise, if a malicious Sub reveals the plaintext of its subscription to the broker, they can also infer the content of I of other Subs. Formally, if both I and I' match τ and the plaintext of I' is known to the broker, then the broker learns I could be the same as I'. Above all, the main issue in previous encrypted pub/sub systems is that the broker not only knows all the matching results, but also stores all the subscriptions.

Based on the above observation, to resist collusion attacks, our basic idea is to (i) organise the brokers in three different domains, (ii) divide the matching operation between tags and subscriptions into three phases, and (iii) enable the brokers in each domain to perform one of the three phases, so as to limit the information learned by each broker. In this case, if the malicious Subs or Pubs collude with one or two types of brokers, they can not infer the subscription of innocent Subs. In the following, we show how the functionalities are broken down.

Our first step is to separate the interests $\mathbf{I}$ and constraint T for each subscription, and send them to two different types of brokers. The type of broker receiving $\mathbf{I}$ is responsible for matching the encrypted interests with tags. We identify this type of brokers as B_I. The broker type that gets T is responsible for evaluating the subscription tree by integrating the match results for all interests with constraint T and forwarding the publications to Subs. This type of brokers is defined as B_T.

However, this step is not enough to resist the collusion attacks. By performing the match between encrypted interests and tags, brokers of type B_I know the match result for each (I, τ) pair. If a broker of type B_I colludes with a malicious Pub or Sub, it could still learn what an innocent Sub is interested in. To address this issue, our second step is to randomly invert some of the interests and add a 'NOT' node in the constraint part on top of the corresponding inverted interest when defining the subscription tree. For instance, in the example subscription in Fig. 3(a), the subscriber changes the interests 'Age $= 18$' and 'Injury=Burn' into 'NOT (Age $\neq 18$)' and 'NOT (Injury $\neq$ Burn)', respectively. The corresponding tree structure is shown in Fig. 3(b). The B_I brokers only get the encrypted interests of a subscription (they do not receive the T part of the subscription). Therefore, even if B_I brokers know τ in plaintext (either by colluding with a malicious Pub or Sub) they cannot learn the real interests of the innocent Sub, since they do not know if there is a 'NOT' node on top of the interest in the corresponding T part of the subscription tree (more details are provided in Section VII).

The matching results between each pair of encrypted $\mathbf{I}$ and $\boldsymbol{\tau}$ are stored in an array and sent to B_T brokers for the subscription evaluation. Specifically, the array contains 1 when the interest matches the corresponding tag, and 0 otherwise. Since B_T brokers store T for each subscription, they know if there is a 'NOT' constraint on top of each leaf node, so they can learn if each interest sent to B_I brokers is inverted or not. However, B_T brokers have no idea what the underlying interest is for each leaf node. Thus, even if a B_T broker colludes with a Pub and/or a Sub, it still cannot learn the interests of innocent Subs. For instance, assume a malicious Pub issues a publication with tags $\{Age : 20, Injury : Burn, Hospital : A\}$ and a B_T broker gets $(1, 1, 0, 0)$ for the subscription shown in Fig. 3. Based on the structure of T, the B_T broker knows the matching result should be $(0, 1, 1, 0)$. However, they cannot determine which two of the tags match the second and the third interests, respectively.

By implementing these two steps, the confidentiality of subscriptions is protected when malicious Subs/Pubs collude with one type of brokers. However, if malicious Pubs or Subs collude with both B_I and B_T, they could infer the innocent Subs' interests, since they would know both the underlying interest for each leaf node and if the interest is inverted or not. In the previous example, a B_I broker knows which tags match or do not match the four interests. Combining this information with T, the colluding entities can infer the plaintext of the matched interests.

To protect the subscription confidentiality of innocent Subs, even when malicious Pubs or Subs collude with both types of brokers, our third step is to randomly permute the interests and involve a third type of broker, B_r, into the matching operation. Specifically, before sending the interests to B_I, the Sub permutes the interests with a Pseudo-Random Permutation (PRP) $\pi : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^n$, and sends the permutation key to a B_r broker. Without the permutation key, even if the malicious Pubs and/or Subs collude with both B_I and B_T brokers, they cannot map the interests in B_I to the constraints stored in B_T. Therefore, they are unable to know which interests are inverted by the Sub. For instance, for the subscription shown in Fig. 3, assume the interests stored in B_I is (I_2, I_4, I_1, I_3) and the matching result obtained by B_I is $(1, 1, 1, 1)$. From T, they know the second and the third leaf nodes are inverted. From the plaintext of the tags, they learn the plaintext of (I_2, I_4, I_1, I_3). However, they do not know which interests stored in the B_I broker are the second and the third leaf nodes of T.

By permuting the interests, the confidentiality of subscriptions is also protected when malicious Pubs and/or Subs collude with B_I and B_r, or B_r, and B_T brokers. In the former collusion scenario, the colluding entities would only recover the order of the interests, yet they cannot learn the structure of T. Likewise, in the latter collusion scenario, they would be able to know the correct order of the matching results and the structure of T, but they cannot learn the order of the interests stored in B_I.

Thus, our scheme is able to protect the subscriptions of innocent Subs as long as at most two types of brokers collude with malicious Pubs and/or Subs. More details on the security

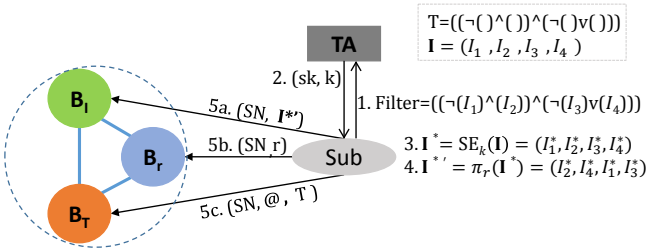


Fig. 4: The Process of Subscription: Each Sub is identified with a unique serial number SN and an address @. Before sending the subscription S to brokers, the Sub encrypts its interests $\mathbf{I}$ into $\mathbf{I}^*$ using SE, and then permutes them into $\mathbf{I}^{*'}$ using PRP π and a random key r.

of our solution will be provided in Section VII.

C. Subscribing and Publishing Procedures

Our proposed solution consists of four different procedures detailed as follows.

Setup. The TA takes as inputs the security parameters λ and κ and generates the public parameters PP and the master key MK for the KP-ABE encryption by running $\texttt{Setup}_{\texttt{KP-ABE}}(\lambda)$. Moreover, it generates the key k for performing match operations based on searchable encryption by running $\texttt{Gen}_{\texttt{SE}}(\kappa)$. k is sent to each enrolled Pub and Sub. MK is kept secure and only known to the TA.

Subscribe. Each Sub is identified by a unique serial number SN and an address @. The brokers use the address to forward the publications to the respective Sub. To subscribe for publications, a Sub performs the following operations:

- First, the Sub defines the subscription $S = (T, \mathbf{I})$ (as shown in Fig. 3(b)). Then, it sends S to the TA in a secure manner (Step 1, Fig. 4).
- The TA takes S as the access structure in the KP-ABE scheme and generates the secret key sk. This key allows the Sub to decrypt the publications that match the subscription S. The TA sends in a secure manner sk to the Sub (Step 2).
- The Sub encrypts each of its interests in $\mathbf{I}$ using the SE scheme: $\mathbf{I}^* = (\texttt{AddToken}_{\texttt{SE}}(k, I_1), \ldots)$ (Step 3).
- The Sub permutes the order of the interests by computing $\mathbf{I}^{*'} = \pi_r(\mathbf{I}^*)$, where r is a random value used as the key for permutation (Step 4).
- Finally, the Sub forwards $(SN, \mathbf{I}^{*'})$ (Step 5a), (SN, r) (Step 5b), and $(SN, @, T)$ (Step 5c) to brokers of type B_I, B_r, and B_T, respectively.

Publish. When publishing a publication, the Pub first generates its ID and defines the set of tags $\boldsymbol{\tau}$ according to its content C. The Pub encrypts $P = (ID, \boldsymbol{\tau}, C)$ as follows:

- Using KP-ABE, the Pub encrypts C to produce $C^* = \texttt{Encrypt}_{\texttt{KP-ABE}}(\texttt{PP}, C, \boldsymbol{\tau})$. Note that we use the set of tags $\boldsymbol{\tau}$ as the attribute set for the KP-ABE encryption. Only the Subs whose subscriptions match against the tags are able to decrypt the publication.

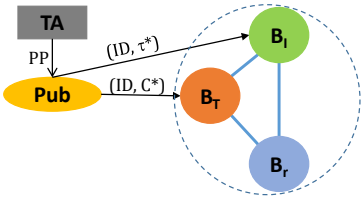


Fig. 5: The Process of Publication: Each publication is identified with a unique ID. When publishing, the Pub encrypts the tags $\boldsymbol{\tau}$ into $\boldsymbol{\tau}^*$ using SE, and encrypts the content C into C^* using KP-ABE.

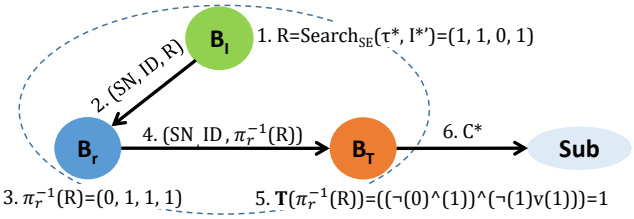


Fig. 6: The Process of Matching and Forwarding: B_I performs the match between interests $\mathbf{I}^{*'}$ and tags $\boldsymbol{\tau}^*$ and gets the matching result R. B_r recovers the order of the result by computing $\pi_r^{-1}(R)$. B_T combines the result with T and gets the final result.

- Afterwards, the Pub encrypts each tag in $\boldsymbol{\tau}$ by computing $\boldsymbol{\tau}^* = \texttt{SearchToken}_{\texttt{SE}}(k, \boldsymbol{\tau})$. This allows the brokers to perform encrypted match operations without accessing the publications and subscriptions in cleartext.
- Finally, as illustrated in Fig. 5, $(ID, \boldsymbol{\tau}^*)$ and (ID, C^*) are sent to B_I and B_T, respectively.

Matching and Forwarding. The matching and forwarding operations are performed in three phases with the cooperation of three types of brokers, as shown in Fig. 6. The operations performed by each type of broker are given below:

- A B_I broker gets $(SN, \mathbf{I}^{*'})$ from the Sub and $(ID, \boldsymbol{\tau}^*)$ from the Pub. First, it matches each interest in $\mathbf{I}^{*'}$ against the corresponding tag in $\boldsymbol{\tau}^*$ by executing $\texttt{Search}_{\texttt{SE}}(\boldsymbol{\tau}^*, \mathbf{I}^{*'})$, and gets the matching result $R = (0/1, \ldots)$ (Step 1). Then, the B_I broker sends (SN, ID, R) to a B_r broker (Step 2). Note that the attribute names in the tags and interests, like 'Injury' and 'Age', are in plaintext or encrypted using a deterministic algorithm. This allows the B_I broker to be able to perform the matching of the tag values with the corresponding interest value.
- For each matching result R received from the B_I broker, the B_r broker retrieves the correct order by running $\pi_r^{-1}(R)$ (Step 3) and forwards $(SN, ID, \pi_r^{-1}(R))$ to a B_T broker (Step 4).
- The B_T broker computes $T(\pi_r^{-1}(R))$, *i.e.,* combines the $0/1$ in R with the AND, OR and NOT operators in T, and gets the final match result (Step 5). If it outputs 1, *i.e.,* $\boldsymbol{\tau}$ satisfies S, C^* is sent to the Sub identified by SN at address @ (Step 6).

Decryption. After receiving the C^*, the Sub uses its secret KP-ABE key related to its subscription to retrieve the content by computing $C = \texttt{Decrypt}_{\texttt{KP-ABE}}(sk, C^*)$.

D. Generalisation

As mentioned before, our system can be generalised to a case where $\mathcal{N}+1$ types of brokers are employed to perform the encrypted filtering and forwarding. One possible generalisation is to set two of types of brokers as B_I and B_T, respectively, and set all the remaining brokers as B_r. Moreover, when subscribing, each Sub has to permute its encrypted interest set $\mathbf{I}^*$ for $\mathcal{N}-1$ times with $\mathcal{N}-1$ different keys, and distribute the keys to the $\mathcal{N}-1$ B_r in different domains. Likewise, when publishing, the match result R obtained by the B_I broker has to be permuted inversely by each B_r broker with the corresponding key. Then, the B_T broker can evaluate the subscription tree to get the final match result. In this case, only when malicious Pubs or Subs collude with all the types of brokers, they can infer other Subs' subscriptions. Otherwise, they cannot learn (i) which interest are inverted if the B_T broker is not part of the colluding set of brokers; (ii) which tags match the interests if the B_I broker does not collude; or (iii) the right order of the match result when one or more B_rs do not collude.

VII. Security Analysis

In this section, we first discuss the security of the publication payload. Second, we analyse the robustness of our solution against collusion attacks. Finally, we discuss the limitations of our scheme and give some countermeasures.

A. The Security of Subscriptions and Publications

In our system, the content of a publication is protected using a lightweight KP-ABE encryption scheme [37]. This scheme has been proven to be INDistinguishable under Chosen Plaintext Attack (IND-CPA) and secure under the Decisional Diffie-Hellman over Elliptic Curve (EC-DDH) assumption. Both subscriptions and tags are encrypted with SUISE [38]. In [38], Hahn and *et al.* prove SUISE achieves IND2-CKA security [42], indicating the security of tags and subscriptions.

B. Collusion Attacks

Our solution is designed to ensure confidentiality of subscriptions against the collusion between malicious Subs/Pubs and curious brokers. We achieve this by splitting the broker functionalities into three different procedures performed by three different types of brokers. Moreover, our system can resist a collusion attack between malicious Subs or Pubs colluding with up to two brokers. In the following, we show the security of subscriptions under different collusion scenarios.

Malicious Subs and Pubs colluding with B_I and B_r brokers. In our solution, B_I brokers store the encrypted interests for each subscription and are responsible for checking if each interest matches a tag. B_r brokers store the permutation key for each subscription and recover the order of each matching result array. Thus, B_I brokers know if there is a match between each encrypted interest and tag pair, and B_r brokes know how the interests are permuted for each subscription. If a malicious Pub colludes with both B_I and B_r brokers, the colluding entities know the publication tags issued by the malicious Pub in plaintext, and they can infer the content of the interests matching against the tags. Similarly, if a malicious Sub colludes with both B_I and B_r brokers, then the colluding entities can infer the interests of the innocent Subs that match the same tags as the interests of the malicious Sub. However, the colluding entities do not have access to the constraint structure T of other innocent Subs. Due to the randomly added 'NOT' gates in T, which are only known to B_T brokers, there is a 50% probability that the real matching result for each interest is opposite to what B_I brokers can infer. In this way, the colluding entities cannot easily infer the real interests of innocent Subs'.

Malicious Subs and Pubs colluding with B_I and B_T brokers. The B_T brokers know the constraint structure T (including the 'NOT' gates) for each subscription and are responsible for getting the final matching result by evaluating the constraints over the matching results between tags and interests that are provided by B_I brokers. However, only B_r brokers are able to recover the correct order of the interests. Note that B_r brokers recover the order of the matching results and forward them to B_T brokers: no attribute names are forward to B_T brokers. Considering that each matching result is an array consisting of 0 or 1, it is still difficult to recover the order of the interests just based on the matching results. For instance, for the subscription shown in Fig. 3, assume the encrypted interests set stored in a B_I broker is $(I_2^*, I_4^*, I_1^*, I_3^*)$, and the matching result with a publication obtained by the B_I broker is $(0, 1, 0, 1)$. A B_T broker will get the array in the right order from the B_r broker, which is $(0, 0, 1, 1)$. By comparing $(0, 0, 1, 1)$ with $(0, 1, 0, 1)$, the order of the interests could be (I_1, I_2, I_3, I_4), (I_2, I_1, I_3, I_4), (I_1, I_2, I_4, I_3), or (I_2, I_1, I_4, I_3). However, the colluding entities cannot determine the correct order. Formally, assume there are m interests in the subscription, meaning a B_I broker will get an m-bit matching result when there is a publication. Assume x out of the m bits are 1. There are $x!$ permutations for the interests whose match results are 1, and $(m-x)!$ permutations for those whose match results are 0. In other words, B_I and B_T brokers could successfully guess the permutations for 1 bits and 0 bits with $\frac{1}{x!}$ and $\frac{1}{(m-x)!}$ probabilities, respectively. To make this probability negligible, the interests in each subscription can be padded with fake interests, such that both $\frac{1}{x!}$ and $\frac{1}{(m-x)!}$ are negligible.

Malicious Subs and Pubs colluding with B_r and B_T brokers. The B_r brokers recover the order of the matching results, and B_T brokers know if each interest is inverted or not. In the example above, the malicious Subs and Pubs colluding with B_r and B_T broker can learn that the interests are permuted into (I_2, I_4, I_1, I_3) by the B_r broker, and I_3 and I_4 match tags. However, they do not know which tags match I_3 and I_4 respectively. This information is only known to the B_I broker. As a result, even if the colluding entities know the

plaintext of the four tags, they do not know which two of them are the real interests of the innocent Subs.

C. Mitigating Guessing Attacks

Although each interest is inverted with 50% possibility, it is not easy to infer whether the interest is real or dummy. However, when there are few interests in a subscription, it might be easier for B_I and B_T brokers to infer the permutation of the interests. To this end, we suggest to pad subscriptions with some dummy interests. Introducing dummy interests also makes it harder to infer the interests of innocent Subs' even when malicious Subs (or Pubs) collude with both B_I and B_r brokers. Investigating the detailed mechanism and the impact of dummy interests on the performance is one of research directions for our future work.

VIII. Analysis of Computation and Communication Costs

In this section, we provide an analysis of computation and communication costs of our solution as well as compare those costs with the schemes listed in Table I.

A. Computation and Communication Costs of Our Solution

This section, we empirically analyse the computation complexities and communication overheads of each entity in our proposed solution.

Let PRF, $HASH$ and PRP be the computation overhead of performing the corresponding operations, and let ABE represents the overhead of using the KP-ABE scheme (see [43] for details of the computation complexity). Table II lists the computation and communication overhead for each entity and each phase in our solution.

Specifically, in the subscription phase, the Sub encrypts all the interests and permutes them, thus the computation complexity of the subscription operation is $m(HASH + PRF) + PRP$, where m is the number of interests in the subscription. Moreover, the communication overhead of this phase is $3|SN| + m\kappa + |\eta| + \kappa + |@| + |T|$ bits[1], where κ is the bit length of the encrypted interests. When issuing a publication, the Pub encrypts the tags and payload with $\mathcal{F}$ and KP-ABE, respectively. The computation cost of a publication is $mPRF + ABE$, and the communication overhead is $2|ID| + m\kappa + |C^*|$ bits, where m is the number of tags in the publication and κ is the size of the encrypted tags in bits. B_I brokers are only responsible for checking if the tags match against the interests and sending the result to B_r brokers. Therefore, the computation complexity and communication overhead for B_I brokers are $mHASH$ and $|SN| + |ID| + m$ bits, respectively. B_r brokers reverse the permutation of the match result and send it to B_T brokers, so their computation complexity and communication overhead are PRP and $|SN| + |ID| + m$ bits, respectively. B_T brokers just evaluate the final output of the subscription tree, and forward the publication payloads to the Sub if the output is 1. Thus, their computation overhead can be negligible and the communication volume is $|C^*|$ when there is a match.

In summary, to ensure confidentiality of tags and interests and support the encrypted matching between them, the computation overhead on the system is $2m(PRF + HASH) + 2PRP$, and the overhead to protect the payload is determined by the applied KP-ABE scheme. In addition to the traffic generated by the publications and subscriptions, the additional communication overhead in our system is generated by the match result R sent from B_I to B_r brokers and the m bits of the permuted result $\pi(R)$ sent from B_r to B_T brokers.

[1]Note that the nonce η can be the same for all the m interests.

B. Comparison with other Pub/Sub Systems

Table III summarises the comparison between our solution and the related work in Table I with respect to the computation cost of encrypted filtering and publication payload encryption, and the communication cost of the related work. Note that at the time of writing this paper, we did not have access to the implementations of the related work schemes. Therefore, the following analysis is based on the details contained in the respective papers.

Computation cost. In terms of the computation overhead, for clarity, we summarise the high-level cryptographic primitives used in other systems for encrypted filtering and payload encryption separately.

To support encrypted filtering, as shown in Table III, most of the pub/sub systems utilise expensive FHE, bilinear paring or other asymmetric encryption primitives. In our work, we use only PRF, hash function and PRP which are significantly more efficient than the FHE and bilinear pairing techniques. The only approach proposed more efficient than our system is the work described in [21]: here the authors use only two primitives including PRF and XOR, while our scheme uses PRF, hash function and PRP.

For the payload encryption, [13], [15], [16], [18]–[21] use symmetric encryption, which is much more efficient than ABE and lattice-based encryption. However, symmetric encryption requires all the Pubs and Subs to share the encryption key. When one of them colludes with the brokers, all the publications will be revealed. By using ABE or lattice-based encryption, Pubs can enforce fine-grained access policy on their publication and avoid key sharing.

Communication cost. In [8]–[10], [16], [19], [20], [23], brokers receive each publication from Pubs and forward it to the Subs whose subscriptions match the publication' tags. Let m be the number of attributes defined in the publication and n be the number of subscriptions matching the publication. Assume $|\tau|$ and $|C^*|$ represent the length of each encrypted tag and the publication payload in bits, respectively. Formally, the communication cost of these schemes is $(n+1)|C^*|+m|\tau|$ bits.

In our approach, there is additional communication between the three types of brokers, *i.e.,* the matching result R. Fortunately, the matching result R is just m-bit long. When there are N subscriptions on B_I, for each publication, B_I (B_r) only needs to send Nm bits to B_r (B_T). In total, the communication cost of our solution is $(n+1)|C^*| + m|\tau| + 2mN$ bits,

TABLE II: Computation and communication costs of our solution.

Operation	Computation Cost	Communication Cost
Subscribe	$m(PRF + HASH) + PRP$	$3\|SN\| + m\kappa + \|\eta\| + \|r\| + \|@\| + \|T\|$ bits
Publish	$mPRF+$ KP-ABE	$2\|ID\| + m\kappa + \|C^*\|$ bits
Match on B_I	$mHASH$	$\|SN\| + \|ID\| + m$ bits
Permutation on B_r	PRP	$\|SN\| + \|ID\| + m$ bits
Evaluation on B_T	Negligible	0 or $\|C^*\|$ bits

PRP, $HASH$ and PRF represent the computation overheads to perform the corresponding operations. We use ABE to represent the overhead to perform the operations for KP-ABE encryption. m is the number of interests/tags in the subscription/publication. κ is the size of the encrypted interest/tag in bits.

TABLE III: Computation and communication cost comparison of pub/sub systems.

Schemes	Computation Cost		Communication Cost
	Primitives for encrypted filtering	Primitives for payload encryption	
Tariq *et al.* [24]	PEKS	CP-ABE	N/A
Ion *et al.* [10]	ElGamal based proxy re-encryption	KP-ABE	$(n+1)\|C^*\| + m\|\tau\|$ bits
Asghar *et al.* [9]	PEKS	CP-ABE	
Yang *et al.* [8]	Bilinear pairing	Dual policy ABE	
Borcea *et al.* [23]	No encryption	Lattice-based proxy re-encryption	
Raiciu *et al.* [18]	Different SE schemes	Symmetric encryption	
Nabeel *et al.* [19]	Pailliar FHE	Symmetric encryption	
Nabeel *et al.* [20]	Pailliar FHE	Symmetric encryption	
Pires *et al.* [16]	Symmetric + Public-key encryptions	Symmetric encryption	
Crescenzo *et al.* [21]	PRF+XOR	Symmetric encryption	$(n+1)\|C^*\| + m\|\tau\| + Nm\|\tau\|$ bits
Rao *et al.* [13] [15]	Different SE schemes	Symmetric encryption	$(\alpha n+1)\|C^*\| + m\|\tau\|$ bits
Our Work	Hash+PRP+PRF	KP-ABE	$(n+1)\|C^*\| + m\|\tau\| + 2mN$ bits

m is the number of attributes, N is the number of subscriptions stored on the broker, and n is the number of subscriptions matching the issued publication. $|\tau|$ and $|C^*|$ represent the encrypted tag and payload lengths in bits, respectively. α is a number greater than 1, which represents the security parameter.

which incurs $2mN$ bits extra overhead than [8]–[10], [16], [19], [20], [23] and traditional pub/sub systems.

On the other hand, the only approaches that incur more overheads than our solution are [21] and [13]. In particular, [21] requires to blind all the interests with different nonces, which are stored on a trusted third-party server. To enable the broker to perform the encrypted filtering, for each subscription, the third-party server has to blind the tags with the same nonces as the interests. That is, for each publication, the third-party server has to send N sets of blinded tags to the broker, which incurs an additional overhead of $Nm|\tau|$ bits.

In [13], subscription privacy is also achieved by cloaking real subscriptions with dummy interests based on a security parameter α. This increases the chance of matching between publications and subscriptions. The redundant publications, forwarded to Subs, also increase the communication overhead in the system. Moreover, the Subs have to send their subscriptions to a trusted third-party server for the cloaking, which also increases the communication overhead in the system.

To summarise, our system can resist against collusion attacks between brokers and malicious Pubs and/or Subs, while incurring computation and communication overheads that are not dramatically more expensive that existing solutions.

IX. Performance Analysis

We have implemented a prototype of our system in C++ and tested its performance. The cryptographic primitives used in SUISE, *e.g.,* HMAC, are implemented using the MIRACL 7.0 library, and the KP-ABE scheme for payload encryption is implemented on top of [37], the C++ code of which is available in [43]. The performance of our system was evaluated on a desktop machine running Intel i5 3.3 GHz 4-core processor with 8GB of RAM. Moreover, we simulated the network communication between any two entities with OMNeT++ [44], and set the bandwidth as 100 Mbps. All the results presented in the following are averaged over 20 runs.

As done in other pub/sub systems [15], [19], [28], the number of tags/interests in the publications and subscriptions varies from 1 to 20, which is derived from the industry-standard benchmark used for measuring the performance of pub/sub systems [10]. As mentioned in Section VII, to ensure that the colluding brokers cannot infer the permutation successfully with a non-negligible probability, it is necessary to pad the subscriptions to a larger size with dummy interests. Thus, in our experiments, we also tested the cases where the number of tags/interests varies from 70 to 116, which ensure that the probabilities of inferring the correct permutation are less than $\frac{1}{2^{128}}$ and $\frac{1}{2^{256}}$, respectively.

A. Encryption of Tags and Interests

Fig. 7 shows the time needed for encrypting publication tags and subscriptions when the number of tags/interests varies from 1 to 116. We can observe that the encryption time in both cases grows linearly with the number of elements. Moreover, the encryption time for the interests is higher than for the tags. The reason is that the Sub performs more operations than the Pub in order to preserve privacy of its subscriptions. Specifically, as described in Section V, on the Pub side, each tag is encrypted using $\mathcal{F}$. In our prototype, $\mathcal{F}$ is implemented as a keyed hash function, SHA-512. However,

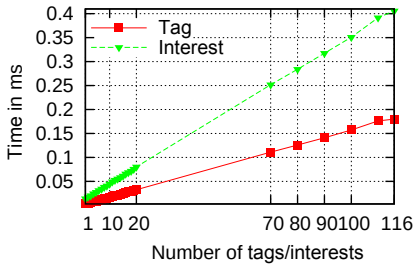


Fig. 7: The time required for encrypting tags and subscriptions when the number of tags and interests varies.

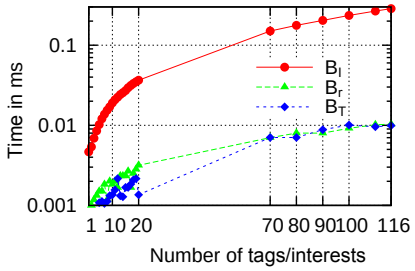


Fig. 8: The time required on each broker for performing their respective operation when the number of tags/interests varies.

for each interest, the Sub first encrypts it with $\mathcal{F}$, and then hashes a nonce by taking the encrypted interest as a key. Moreover, all the interests have to be shuffled, *i.e.,* permuted, before sending to the brokers. In the experiment, the shuffle operation is implemented with Fisher-Yates algorithm [45].

B. Encrypted Matching

Matching the publications to the subscriptions is performed with the collaboration of three types of brokers. As mentioned in Section VI, a B_I broker checks if the encrypted tags match against the encrypted interests, and then this B_I broker sends the result array to a B_r broker. The B_r broker recovers the order of the array and sends it to a B_T broker. Finally, the B_T broker evaluates the result and forwards the publication to the corresponding Subs.

Fig. 8 shows the time required for each broker to perform their respective operation when the number of tags/interests varies. We can see that the matching time on B_I brokers increases linearly with the increase in the number of tags/interests. When there are 116 elements, it takes less than 0.3 ms on a B_I broker to check all the tags and interests. The permutation time on a B_r broker and the evaluation time on a B_T broker also increase with the number of elements. However, the time required by the B_r and B_T brokers is around 0.01 ms when there are 116 elements, which is negligible.

C. Payload Encryption and Decryption

In our implementation, the publication payload is encrypted with AES-CBC using a randomly generated symmetric key. Then, the KP-ABE scheme is used to encrypt the symmetric key. The KP-ABE scheme is affected by the number of tags,

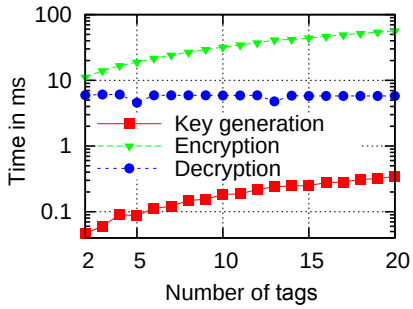


Fig. 9: The time for encrypting and decrypting the payload using KP-ABE in publications with a different number of tags. The graph also provides the time required for generating the KP-ABE key.

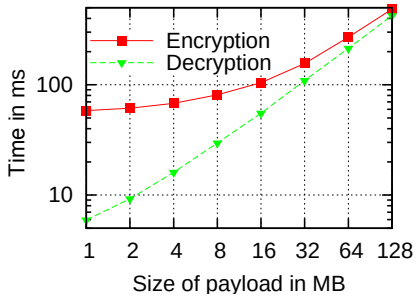


Fig. 10: The time of encrypting the payload of the publications with different payload sizes.

while the AES-CBC algorithm is affected by the payload size. We show the effects of the number of tags and the payload size on publication encryption and decryption in Fig. 9 and Fig. 10, respectively.

Fig. 9 shows the key generation time for the Sub's interests performed on the TA, the payload encryption time on the Pub, and the decryption time on the Sub. In this experiment, the payload size is fixed to 1 MB, and the number of tags varies from 2 to 20. Note that the dummy tags and interests are not involved in the access policy and decryption keys. Both the encryption and decryption times shown in Fig. 9 include the time of encrypting/decrypting the payload with AES-CBC and the time of encrypting and decrypting the symmetric key with KP-ABE. We can see that both the encryption and key generation times go up linearly with an increase in the number of elements. However, the effect of tags on payload decryption is not evident, thanks to the lightweight construction of the KP-ABE implementation used in our system.

Fig. 10 shows the effect of payload sizes on the encryption and decryption time. For this experiment, we fixed the number of tags for KP-ABE to 20. As we can see in the figure, both the encryption and decryption time rises linearly with the payload size. When the payload size is 128 MB, the payload encryption and decryption take up to 485.5 ms and 424.6 ms, respectively.

D. Performance Tradeoff

In this experiment, we show the performance tradeoff to achieve R1, R2, R3, and R4 in our solution when there are

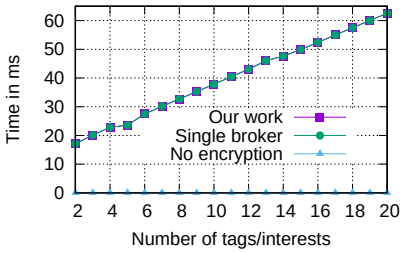


Fig. 11: The end to end latency comparison among our solution and two baseline scenarios. In 'Single broker' scenario, the publications and subscriptions are encrypted in the same way as our work, but only one broker is used. In 'No encryption' scenario, all the data is in plaintext and only one broker is used.

different numbers of tags/interests. The payload size is set to 1 MB.

To achieve R1, R2, and R3, our solution leverages KP-ABE and SE schemes to encrypt the publications and subscriptions. To show the performance tradeoff to achieve R1, R2, and R3, we compare the end-to-end latency of forwarding a publication to a subscriber in our solution with that of a baseline scenario, called 'No encryption', where both the publication and the subscription are in plaintext. To achieve R4, our solution randomly adds 'NOT' gates into the subscription tree, permutes the order of the interests, and distributes the tasks to different brokers. To show the cost of resisting collusion attacks, we compare the end-to-end latency of our solution with another baseline scenario 'Single broker', where only one broker is used to perform the encrypted filtering as done in a typical setting. Of course, in such a scenario, no privacy can be guaranteed if the broker colludes with malicious Pubs or Subs.

The results are shown in Fig. 11. We can see that the latency of 'Our work' almost overlaps with the 'Single broker' scenario, whereas there is a big gap between 'Our work' and the 'No encryption' scenario. These results show that the extra cost of resisting collusion attacks in our solution (*i.e.,* using three different types of brokers) is negligible when compared with a typical single-broker scenario. The main computation overhead in our solution is added by the data encryption to ensure data confidentiality and the de-coupling property. Specifically, when there are 20 tags and interests, the end-to-end latencies for 'Our work', 'Single broker', and 'No encryption' are 62.521 ms, 62.518 ms and 0.012 ms, respectively. In this case, the techniques to achieve R1, R2, and R3 add a $5208.8\times$ overhead when compared to a plaintext solution.

X. Conclusions and Future Work

In pub/sub systems, publications are disseminated to interested subscribers through a set of brokers. These brokers are able to collect sensitive information by accessing publications' tags and subscribers' interests. Although existing solutions enable encrypted matching, they cannot protect subscriptions of innocent subscribers if malicious subscribers (or publishers) collude with untrusted brokers. To address this issue, in this article, we propose a solution that uses three different types of brokers and splits the matching operation into three phases, where each phase is executed by a different type of broker. Even in the case of malicious subscribers (or publishers) colluding with up two different types of brokers, they are unable to infer the subscriptions of innocent subscribers.

In this work, the brokers are assumed to follow the protocol honestly. In practice, compromised brokers may tamper the data actively. As future work, we aim to investigate approaches to identify the malicious behaviour of brokers, such as sending publications to unintended subscribers or not forwarding the matched publications to intended subscribers. In general, our goal is to make brokers accountable for actions they perform.

The SE scheme (*i.e.,* SUISE) used in our system only supports equality check between encrypted tags and interests. For future work, we will also consider to support complex operations, such as range queries.

Acknowledgements

This research is supported by STRATUS (Security Technologies Returning Accountability, Trust and User-Centric Services in the Cloud), a project funded by the Ministry of Business, Innovation and Employment (MBIE), New Zealand.

References

[1] S. Cui, S. Belguith, P. D. Alwis, M. R. Asghar, and G. Russello, "Malicious entities are in vain: Preserving privacy in publish and subscribe systems," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, Aug 2018, pp. 1624–1627.

[2] D. E. Bakken, A. Bose, C. H. Hauser, D. E. Whitehead, and G. C. Zweigle, "Smart generation and transmission with coherent, real-time data," *Proceedings of the IEEE*, vol. 99, no. 6, pp. 928–951, 2011.

[3] C. Esposito, M. Ciampi, and G. De Pietro, "An event-based notification approach for the delivery of patient medical information," *Information Systems*, vol. 39, pp. 22–44, 2014.

[4] M. Cinque, C. Di Martino, and C. Esposito, "On data dissemination for large-scale complex critical infrastructures," *Computer Networks*, vol. 56, no. 4, pp. 1215–1235, 2012.

[5] I. M. Delamer and J. L. M. Lastra, "Service-oriented architecture for distributed publish/subscribe middleware in electronics production," *IEEE Transactions on Industrial Informatics*, vol. 2, no. 4, pp. 281–294, 2006.

[6] "Google cloud pub/sub," https://cloud.google.com/pubsub, last accessed: November 27, 2018.

[7] "Yahoo data breach," https://www.theguardian.com/technology/2016/dec/14/yahoo-hack-security-of-one-billion-accounts-breached, 2016, last accessed: November 27, 2018.

[8] K. Yang, K. Zhang, X. Jia, M. A. Hasan, and X. S. Shen, "Privacy-preserving attribute-keyword based data publish-subscribe service on cloud platforms," *Information Sciences*, vol. 387, pp. 116–131, 2017.

[9] M. R. Asghar, A. Gehani, B. Crispo, and G. Russello, "PIDGIN: Privacy-preserving interest and content sharing in opportunistic networks," in *Proceedings of the 9th ACM symposium on information, computer and communications security*. ACM, 2014, pp. 135–146.

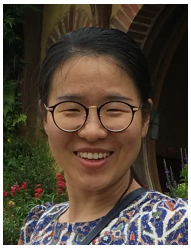
[10] M. Ion, G. Russello, and B. Crispo, "Design and implementation of a confidentiality and access control solution for publish/subscribe systems," *Computer networks*, vol. 56, no. 7, pp. 2014–2037, 2012.

[11] C. Esposito and M. Ciampi, "On security in publish/subscribe services: A survey," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 966–997, 2015.

[12] B. Shand, P. Pietzuch, I. Papagiannis, K. Moody, M. Migliavacca, D. Eyers, and J. Bacon, "Security policy and information sharing in distributed event-based systems," *Reasoning in Event-Based Distributed Systems*, pp. 151–172, 2011.

[13] W. Rao, L. Chen, and S. Tarkoma, "Toward efficient filter privacy-aware content-based pub/sub systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 11, pp. 2644–2657, 2013.
[14] E. Onica, P. Felber, H. Mercier, and E. Rivière, "Confidentiality-preserving publish/subscribe: A survey," *ACM Computing Surveys (CSUR)*, vol. 49, no. 2, p. 27, 2016.
[15] W. Rao, L. Chen, M. Yuan, S. Tarkoma, and H. Mei, "Subscription privacy protection in topic-based pub/sub," in *International Conference on Database Systems for Advanced Applications*. Springer, 2013, pp. 361–376.
[16] R. Pires, M. Pasin, P. Felber, and C. Fetzer, "Secure content-based routing using Intel Software Guard Extensions," in *Middleware 2016*. ACM, 2016, p. 10.
[17] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *S&P 2000*. IEEE Computer Society, 2000, pp. 44–55.
[18] C. Raiciu and D. S. Rosenblum, "Enabling confidentiality in content-based publish/subscribe infrastructures," in *Second International Conference on Security and Privacy in Communication Networks and the Workshops, SecureComm 2006, Baltimore, MD, Aug. 28 2006 - September 1, 2006*. IEEE, 2006, pp. 1–11.
[19] M. Nabeel, N. Shang, and E. Bertino, "Efficient privacy preserving content based publish subscribe systems," in *Proceedings of the 17th ACM symposium on Access Control Models and Technologies*. ACM, 2012, pp. 133–144.
[20] M. Nabeel, S. Appel, E. Bertino, and A. Buchmann, "Privacy preserving context aware publish subscribe systems," in *International Conference on Network and System Security*. Springer, 2013, pp. 465–478.
[21] G. D. Crescenzo, J. Burns, B. A. Coan, J. L. Schultz, J. R. Stanton, S. Tsang, and R. N. Wright, "Efficient and private three-party publish/subscribe," in *NSS 2013*, ser. Lecture Notes in Computer Science. Springer, 2013, pp. 278–292.
[22] S. Choi, G. Ghinita, and E. Bertino, "A privacy-enhancing content-based publish/subscribe system using scalar product preserving transformations," in *International Conference on Database and Expert Systems Applications*. Springer, 2010, pp. 368–384.
[23] C. Borcea, Y. Polyakov, K. Rohloff, G. Ryan *et al.*, "PICADOR: End-to-end encrypted publish–subscribe information distribution with proxy re-encryption," *Future Generation Computer Systems*, vol. 71, pp. 177–191, 2017.
[24] M. A. Tariq, B. Koldehofe, and K. Rothermel, "Securing broker-less publish/subscribe systems using identity-based encryption," *IEEE transactions on parallel and distributed systems*, vol. 25, no. 2, pp. 518–528, 2014.
[25] G. D. Crescenzo, B. A. Coan, J. L. Schultz, S. Tsang, and R. N. Wright, "Privacy-preserving publish/subscribe: Efficient protocols in a distributed model," in *SETOP 2013*, ser. Lecture Notes in Computer Science. Springer, 2014, pp. 114–132.
[26] E. Onica, P. Felber, H. Mercier, and E. Rivière, "Efficient key updates through subscription re-encryption for privacy-preserving publish/subscribe," in *Proceedings of the 16th Annual Middleware Conference*. ACM, 2015, pp. 25–36.
[27] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *EUROCRYPT 1999*, ser. Lecture Notes in Computer Science, J. Stern, Ed., vol. 1592. Springer, 1999, pp. 223–238.
[28] M. Nabeel and E. Bertino, "Attribute based group key management," *Trans. Data Privacy*, vol. 7, no. 3, pp. 309–336, 2014.
[29] W. K. Wong, D. W.-l. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. ACM, 2009, pp. 139–152.
[30] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-Policy Attribute-Based Encryption," in *S&P 2007*. IEEE Computer Society, 2007, pp. 321–334.
[31] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *EUROCRYPT 2004*, ser. Lecture Notes in Computer Science, C. Cachin and J. Camenisch, Eds., vol. 3027. Springer, 2004, pp. 506–522.
[32] C. Dong, G. Russello, and N. Dulay, "Shared and searchable encrypted data for untrusted servers," in *DBSec 2008*, ser. Lecture Notes in Computer Science, vol. 5094. Springer, 2008, pp. 127–143.
[33] V. Costan and S. Devadas, "Intel SGX explained," *IACR Cryptology ePrint Archive*, vol. 2016, p. 86, 2016.
[34] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures," in *CCS 2007*. ACM, 2007, pp. 195–203.
[35] J. Alderman, N. Farley, and J. Crampton, "Tree-based cryptographic access control," in *ESORICS 2017*. Springer, 2017, pp. 47–64.
[36] A. Castiglione, A. D. Santis, and B. Masucci, "Key indistinguishability versus strong key indistinguishability for hierarchical key assignment schemes," *IEEE Trans. Dependable Sec. Comput.*, vol. 13, no. 4, pp. 451–460, 2016.
[37] X. Yao, Z. Chen, and Y. Tian, "A lightweight attribute-based encryption scheme for the internet of things," *Future Generation Comp. Syst.*, vol. 49, pp. 104–112, 2015.
[38] F. Hahn and F. Kerschbaum, "Searchable encryption with secure and efficient updates," in *SIGSAC 2014*. ACM, 2014, pp. 310–320.
[39] M. R. Asghar, G. Russello, B. Crispo, and M. Ion, "Supporting complex queries and access policies for multi-user encrypted databases," in *CCSW 2013*. ACM, 2013, pp. 77–88.
[40] S. Cui, M. R. Asghar, S. D. Galbraith, and G. Russello, "P-McDb: Privacy-preserving search using multi-cloud encrypted databases," in *CLOUD 2017*. IEEE Computer Society, 2017, pp. 334–341.
[41] F. Buccafurri, G. Lax, R. A. Sahu, and V. Saraswat, "Practical and secure integrated PKE+PEKS with keyword privacy," in *SECRYPT 2015*. SciTePress, 2015, pp. 448–453.
[42] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *CCS 2012*, T. Yu, G. Danezis, and V. D. Gligor, Eds. ACM, 2012, pp. 965–976.
[43] "A lightweight Attribute-Based Encryption scheme in c++," https://github.com/ikalchev/kpabe-yct14-cpp, last accessed: November 27, 2018.
[44] "OMNeT++ discrete event simulator," https://www.omnetpp.org, last accessed: November 27, 2018.
[45] R. A. Fisher, F. Yates *et al.*, "Statistical tables for biological, agricultural and medical research," *Statistical tables for biological, agricultural and medical research*, no. 3, 1949.

VITAE



Shujie Cui is a Ph.D. Candidate in the School of Computer Science, The University of Auckland, New Zealand, and a Research Assistant in the Large-Scale Data & Systems (LSDS) group in the Department of Computing at Imperial College London, UK. She received her joint M.Sc. degree in Computer Science from Shandong University, China and the University of Luxembourg, Luxembourg in 2013. She obtained her B.Sc. degree from Shandong University, China in 2011. After obtaining her M.Sc. degree, she worked in the Department of Computer Science at City University of Hong Kong as a Research Associate. Her research interests include applied cryptography, security and privacy in cloud computing and distributed systems.

Sana Belguith is a Lecturer in the School of Computing, Science and Engineering at the University of Salford, Manchester, UK. Previously, she was a Post-Doctoral Researcher in the School of Computer Science at The University of Auckland, New Zealand. She received her engineering degree in Computer Science from the National Engineering School of Tunisia, in 2012 and her Ph.D. degree from the Tunisia Polytechnic School, Tunisia in 2017. As part of her Ph.D. programme, she was a Visiting Fellow at Télécom SudParis, France. Her major research interests include applied cryptography, distributed systems security, privacy enhancing techniques, access control, attribute-based encryption, and searchable encryption.

Pramodya De Alwis is a student at the University of Auckland completing a graduate diploma in Computer Science. Upon completing his civil engineering honours degree in 2016, he decided to study computer science and follow his passion. Currently, he is working as a backend database engineer while also working on two start ups he has co-founded. His interests include cloud computing, machine learning, and data science.

Muhammad Rizwan Asghar is a Senior Lecturer in the School of Computer Science at The University of Auckland in New Zealand. Previously, he was a Post-Doctoral Researcher at international research institutes including the Center for IT-Security, Privacy, and Accountability (CISPA) at Saarland University in Germany and CREATE-NET in Trento Italy. He received his Ph.D. degree from the University of Trento, Italy in 2013. As part of his Ph.D. programme, he was a Visiting Fellow at the Stanford Research Institute (SRI), California, USA. He obtained his M.Sc. degree in Information Security Technology from the Eindhoven University of Technology (TU/e), The Netherlands in 2009. His research interests include access control, cybersecurity, privacy, and consent management.

Giovanni Russello is an Associate Professor in the School of Computer Science at the University of Auckland, New Zealand. He is also the head of the school. He received his M.Sc.(summa cum laude) degree in Computer Science from the University of Catania, Italy in 2000, and his Ph.D. degree from the Eindhoven University of Technology (TU/e) in 2006. After obtaining his Ph.D. degree, he moved to the Policy Group in the Department of Computing at Imperial College London, UK. His research interests include policy based security systems, privacy and confidentiality in cloud computing, smartphone security, and applied cryptography. He has published more than 60 research articles in these research areas and has two granted US Patents in smartphone security. He is an IEEE member.