Cybersecurity

RESEARCH **Open Access**

An efficient confidentiality protection solution for pub/sub system

Jinglei Pei[1], Yuyang Shi[1], Qingling Feng[1], Ruisheng Shi[1,2*], Lina Lan[1,2], Shui Yu[3], Jinqiao Shi[1,2] and Zhaofeng Ma[1,2]

Abstract
Publish/subscribe(pub/sub) systems are widely used in large-scale messaging systems due to their asynchronous and decoupled nature. With the population of pub/sub cloud services, the privacy protection problem of pub/sub systems has started to emerge, and events and subscriptions are exposed when executing event matching on untrustworthy cloud brokers. However, as the number of subscriptions increases, the effectiveness of the previous confidentiality protection approaches declines drastically. In this paper, we propose SBM (scalable blind matching), an effective confidentiality protection scheme for pub/sub systems. To the best of our knowledge, SBM is the first scheme that applies order-preserving encryption algorithm to protect the system's confidentiality and ensure its scalability. In this scheme, SBM-I is highly effective in subscription matching but is unable to achieve ideal security IND-OCPA, whereas SBM-II is suggested to ensure system security and SGX is used to reduce interaction and boost ciphertext matching performance. The experiment demonstrates that this method has better matching performance compared to others: the average matching time of SBM-I is 3–4 orders of magnitude faster than the matching algorithm MP and SGX-based algorithm SCBR when the number of subscriptions is 500,000, and the average matching time of SBM-II is 40 times faster than MP and 24 times than SCBR.

Keywords Pub/sub, Confidentiality, Privacy protection, SGX, Scalability

Introduction

Publish/subscribe (pub/sub) communication architecture has been widely used in large-scale information transmission systems, such as smart buildings (Kumar et al. 2019), e-health systems (Ion et al. 2012), intelligent traffic monitoring systems (Nabeel et al. 2013), and stock monitoring systems Ding et al. (2020), due to its ability to provide complete decoupling in time, space, and synchronization between communication entities (Eugster et al. 2003).

Pub/sub cloud services (Amazon 2022; Google 2022; Microsoft 2022) are widely used with the advent of cloud computing. As shown in Fig. 1, an increasing number of developers choose to externalize services, such as event matching and routing, to public infrastructures, rather than build services themselves. This strategy accelerates time-to-market, lowers operational costs, and brings huge economic advantages to developers.

A pub/sub system usually consists of three participants: (1) publishers which publish events to the system; (2) subscribers that register interested subscriptions to the system; (3) brokers that serve as pub/sub middleware and are responsible for subscription storage, event matching and routing. The system workflow is shown in Fig. 2.

*Correspondence:
Ruisheng Shi
shiruisheng@bupt.edu.cn
[1] Beijing University of Posts and Telecommunications, Beijing 100876, China
[2] Key Laboratory of Trustworthy Distributed Computing and Service (BUPT), Ministry of Education, Beijing, China
[3] School of Computer Science, University of Technology Sydney, Sydney, NSW, Australia

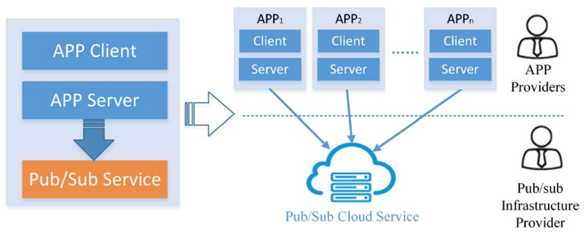

Fig. 1 The evolution of pub/sub application architecture

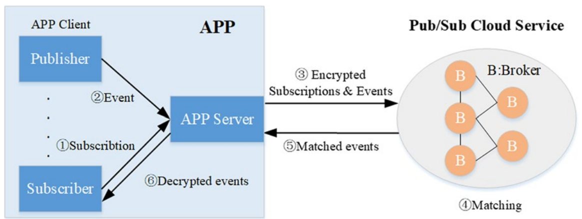

Fig. 2 Pub/sub cloud service privacy protection schemes

1. As app clients, publishers and subscribers send subscriptions and events to the app server respectively.
2. Subscriptions and events are encrypted on the app server and then sent to the untrustworthy broker.
3. The broker performs event matching and routing, then return matched events to the app server.
4. The server decrypts matched events and then returns them to the corresponding subscriber.

However, the development of pub/sub cloud service is accompanied by serious data privacy issues. Cloud services deployed in the public domain are unreliable and attackable, which can lead to the disclosure of sensitive user data and even significant financial losses. To solve the problem, some privacy protection schemes (Kumar et al. 2019; Ion et al. 2012; Raiciu and Rosenblum 2006; Shikfa et al. 2009; Choi et al. 2010; Pal et al. 2012; Tariq et al. 2013; Pires et al. 2016; Barazzutti et al. 2015; Arnautov et al. 2018) have been proposed. Some of the current approaches are based on the security of cryptographic algorithms, while many researchers utilize SGX to safeguard the system's confidentiality.

With the increasingly widespread use of pub/sub systems, the number of subscriptions is growing to tens of millions. However, none of current approaches can meet the performance requirements of a large-scale pub/sub system. Their performance degrades significantly when the number of subscriptions grows. The underlying reason is that these methods must compare the events with encrypted subscriptions item by item when performing event matching. So it is vital to design an efficient privacy protection scheme for large-scale pub/sub systems so that the broker can provide effective event matching and routing while only knowing the events and subscription ciphertexts.

In plaintext matching of events, subscriptions are usually stored by an efficient data structure (Qian et al. 2014; Ji and Jacobsen 2018; Qian et al. 2014) and the mismatched ones are filtered out directly, thus greatly reducing the number of subscriptions should be compared and decreasing the time complexity of the algorithm. Inspired by it, we propose an efficient ciphertext matching scheme, Scalable Blind Matching (SBM).

In particular, we first propose SBM-I based on an order-preserving encryption (OPE) algorithm without index structure. In this scheme, the event and subscription ciphertexts retain the order of original plaintexts, so the efficiency of matching is consistent with that of plaintext matching. But if the plaintexts are evenly distributed, the attacker can obtain at least half of the plaintext bit letters and cannot reach the ideal security IND-OCPA, which denotes indistinguishability under ordered chosen plaintext attacks. To solve the security problems of SBM-I, we update the OPE algorithm used in the scheme and choose the more secure index-based OPE algorithm, MOPE. However, the frequent interaction between client and server in this scheme will reduce the matching efficiency, so SBM-II introduces SGX to reduce the interaction and improve the matching performance. In general, SBM-II performs ciphertext storage and query by MOPE algorithm, and SGX is responsible for decrypting the ciphertext, to offer effective secrecy protection.

The contributions of our work are summarized as follows:

1. We first apply OPE algorithm to the SBM-I to narrow the range of subscription sets that should be compared and make the system scalable.
2. SBM-II is proposed to satisfy the ideal security IND-OCPA. It is based on OPE with index structure to ensure confidentiality and uses SGX to improve the performance of the scheme. In addition, in SBM-II, we construct a new encoding algorithm for the subscription cipher stored in the balanced binary tree to support the matching of subscriptions with different constraint types.
3. Experiments are designed to evaluate the two schemes, and the experimental results show that SBM has higher performance than the matching algorithm based on Paillier homomorphic encryption MP and SGX-based algorithm SCBR.

The rest of the paper is organized as follows. Section "Related work" gives the related work to the paper. System and threat models are described in "Preliminaries" section. Section "Efficient confidentiality protection schemes" illustrates the design and implementation of our schemes SBM-I and SBM-II. Section "Evaluation" shows the results of our experiments. Finally, conclusion and outlook are given in "Conclusion" section.

Related work

Confidentiality protection schemes

The security requirements for pub/sub systems were first proposed by Wang et al. (2002), who defined these requirements as authentication, integrity, confidentiality, and availability. Despite their detailed definition of event confidentiality and subscription confidentiality, they did not propose a method to achieve them.

Current research on confidentiality protection schemes can be divided into two categories. One is ciphertext matching schemes based on cryptographic algorithms to enable brokers to perform event matching without decrypting events and subscriptions. The other is to use the trusted execution environment SGX (software guard extensions) provided by Intel. SGX encapsulates the application's security operations in an Enclave container, where the confidentiality and integrity of code and data are guaranteed, and events and subscriptions are matched in the Enclave created for execution. The following are described separately.

Cryptographic-based schemes

Choi et al. (2010) used an asymmetric scalar-product preserving encryption (ASPE) algorithm for ciphertext matching. ASPE was proposed by Wong et al. (2009) based on the k-nearest neighbor(KNN) algorithm over an encrypted database. The core idea of the scheme is to represent the event attribute values and subscription constraint values as coordinates of points on a multidimensional space, then encrypt the event and subscription coordinates with the ASPE algorithm and determine whether the event and subscription match only by comparing the distance between the encrypted subscription coordinates and the encrypted event coordinates. This method enables complicated type subscriptions, including equal-value, non-equal-value, and interval subscriptions (aside from strings), but its security is weak and can only withstand ciphertext-only attacks.

Ion et al. (2012) proposed to use a multi-user searchable data encryption (SDE) (Dong et al. 2011) algorithm to encrypt events and subscriptions. Before subscription encryption, numerical subscription constraints are represented as access trees (Bethencourt et al. 2007). The access tree consists of leaf nodes and non-leaf nodes. Each leaf node is represented by an attribute and the value of the attribute constraint after SDE encryption. Each non-leaf node is a threshold gate, consisting of a threshold value and its children nodes. The threshold value indicates the minimum number of children nodes needed to satisfy this non-leaf node when matching. If each attribute value in the encrypted event satisfies the access tree corresponding to the same attribute in the subscription, the event matches the subscription. This scheme is based on the ElGmal algorithm, which is more secure than the ASPE algorithm, but less efficient in event matching.

Nabeel et al. (2013) proposed to implement ciphertext matching in a context-based pub/sub system with a modified Paillier homomorphic encryption scheme (Paillier 1999). First, the security parameters encrypted with the modified Paillier algorithm are distributed by the context manager for publishers and subscribers. Then the event and subscription are encrypted under their respective security parameters, and the matching is performed by simply multiplying the subscription ciphertext with the event ciphertext, at which time the homomorphic additive nature of the Paillier algorithm is used to eliminate some security parameters. Finally, the modified Paillier algorithm is used to decrypt the remaining result and determine whether the event and subscription match according to the final result. The solution is theoretically secure and has good matching performance, but it cannot support matching of string type subscriptions, which greatly limits its practical use.

Barazzutti et al. (2015) proposed to use Bloom filters (Bloom 1970) to alleviate the performance problem of ciphertext matching and improving the matching efficiency by reducing the subscription space. However, their scheme can only filter out mismatched subscriptions containing equivalence types, other than that, other types of subscriptions still need to perform ciphertext matching with the ASPE scheme.

Borcea et al. (2017) propose PICADOR, which applies a lattice encryption-based proxy re-encryption scheme that is capable of resisting adversary attacks with quantum computing devices. The Proxy Re-Encryption (PRE) scheme can convert encrypted events into subscriptions that can only be decrypted by approved subscribers.

Gaballah et al. (2021) proposed 2PPS, which introduced distributed point function (DPF) based secret sharing on top of a searchable encryption mechanism, distributing the function of the broker across multiple servers so that privacy is guaranteed as long as at least one server does not collude with the adversary. However, once a malicious server refuses to reply, the protocol execution cannot be continued and is less available.

The basic strategy behind the cryptography-based privacy protection scheme is to implement a ciphertext matching scheme that first encrypts events and subscriptions and then performs matching for each encrypted event with subscriptions one by one. The time complexity of matching is proportional to the number of subscriptions in the system, and matching is not scalable. As the number of subscriptions in the system increases, the performance of the system gets worse.

SGX-based schemes

In 2016, Pires et al. (2016) proposed using SGX to implement secure routing in pub/sub systems. The main idea is to encrypt events and subscriptions outside the Enclave, decrypt them inside the Enclave with the help of TEE provided by the Enclave, and then construct efficient indexes for subscriptions to achieve fast matching. However, the subscriber in this scheme must first send the subscription to the publisher, which violates the decoupling property of the pub/sub system.

In 2018, Arnautov et al. (2018) proposed PUBSUB-SGX. It consists of two main components: the Load Balancer/Proxy and the Matcher. The Load Balancer is responsible for sending events and subscriptions to the Matcher in a balanced manner, and the Matcher deploys multiple broker nodes to perform the matching of events and subscriptions. Thus, PubSub-SGX (Amazon 2022) expands on the capabilities of SCBR by allowing for parallelization and scalability. In this scheme, the balancer and matcher should run different enclaves, but this paper does not describe how they securely interact.

In 2021, Wang et al. (2021) proposed MagikCube, which consists of four components: publisher, subscriber, broker, and authentication service. The scheme also leverages the isolation feature of SGX enclave to ensure that all sensitive operations are agnostic to the adversary. And the certificate service will periodically authenticate the broker to ensure that it is running on a trusted SGX platform. The solution does not require direct communication between subscribers and publishers, ensuring the decoupling property.

Experimental results show that the SGX-based scheme (Pires et al. 2016) performs more efficiently than the scheme using cryptographic algorithms (ASPE Wong et al. (2009)) because the plaintext matching is performed inside SGX and does not require complex operations like ciphertext matching. However, due to the limited memory of Enclave container, when the amount of stored data is large or the occupied memory is large (more than 90 MB) (Pires et al. 2016), page misses and cache misses are prone to occur, and then Enclave needs to interact with system memory to transfer in and out cache or pages, which will increase the overhead additionally. The matching performance of the SGX-based scheme is closely related to the existing hardware performance and is currently not applicable to the processing of large-scale subscriptions.

Order preserving encryption

In the study of order-preserving encryption algorithms (OPE), the existing schemes can be classified into schemes without index structure and schemes based on index structure (Guo et al. 2018). The OPE scheme without index structure means that the encrypted ciphertext directly retains the original plaintext order. An index-based OPE scheme encrypts plaintext data using a common encryption scheme (e.g., AES, DES) and creates an order-preserving index structure that can be used to compare the order of ciphertexts with each other.

The security of existing OPE schemes without index structure is not high, and many of them cannot achieve the desired security of OPE. For example, Agrawal et al. (2004) constructed the first OPE scheme OPES, which is based on the idea of randomly selecting P data in the global space with the target distribution provided by the user and then sorting these data to obtain the key table T. The ciphertext of data d_i in the data fetching space D is $c_i = T[i]$. Since the data in the key table T are sorted, the ciphertext retains the plaintext order. However, this mechanism needs to model the plaintext first when encrypting, so the user has to know all the plaintexts in advance. Besides, Agrawal et al. do not define security for OPE schemes, nor do they give a strict security analysis of their OPES schemes.

Boldyreva et al. (2009) constructed the first provably secure OPE scheme. It was constructed based on the natural relationship between random order-preserving function(ROPF) and the hypergeometric metric probability distribution. Subsequently, it is shown to satisfy ROPF security but does not achieve ideal security.

Index structure-based OPE schemes that can achieve ideal security usually require multiple rounds of interaction between the client and the server. For example, Popa et al. (2013) constructed the first OPE scheme that can achieve ideal security. It uses deterministic encryption techniques such as AES and other symmetric encryption algorithms to encrypt the data, a mutable order-preserving encoding (MOPE) algorithm to encode the plaintext, and stores the ciphertext data and the plaintext encoding in the OPE table, and the ciphertext data are also stored in the OPE tree according to the encoding size. When inserting new data, it requires $O(\log n)$ rounds of interaction between the user and the server to determine the position of the data in the tree and insert the data into the tree, generating its order-preserving index. When querying the data, it is necessary to insert and then compare its index value. In general, this frequent interaction reduces the efficiency of the system in terms of insertion and querying. In addition, this scheme is only suitable for single-user scenarios.

Liang et al. (2020) constructed MSOPE based on Popa et al.'s idea for multi-user scenarios, which can provide a higher security guarantee than the ideal security, like indistinguishability under multi-source ordered chosen plaintext attack (IND-MSOCPA), which can achieve resistance to privacy leakage attacks(guaranteeing that the ciphertext will not reveal information other than the order), frequency analysis attacks, and equivalence speculation attacks. The insertion and query idea is similar to Popa et al.'s scheme, which requires multiple interactions between users and the server.

Preliminaries

In this section, we focus on the pub/sub system model and the threat model.

System model

In the pub/sub system, the events matched are forwarded by the broker node to the corresponding subscribers. To represent the matching and routing of events and subscriptions, we define the data model in the pub/sub

Events An event e published by a publisher consists of several attributes A. For example, an event $e = \{A_1, A_2, \ldots, A_k\}$ with k attributes. Each attribute is represented by a triple (*type*, *name*, *value*) Carzaniga et al. (2001), where *type* represents the type of data; *name* represents the name of the attribute; and *value* represents the value taken on the attribute. For example, in the stock quotation system, IBM publishes an event containing two attributes $e1 = \{(string, name, \text{“}IBM''), (int, price, 128)\}$, then the event indicates that the current stock price of IBM is $128.

Subscriptions A subscription s registered by a subscriber consists of the conjunction of several constraints P. For example, a subscription s containing j constraints $s = \{P_1, P_2, \ldots, P_j\}$. Each constraint is represented by a triple (*name*, *Op*, *value*), where *name* denotes the name of the constraint attribute; *Op* denotes the operator, which includes operations of equal, unequal, and interval types for numeric attributes and operations of prefix and suffix types for string attributes; and *value* denotes the value on the constraint attribute. For example, an investor sends a subscription s_1 with two constraints $s_1 = \{(name, =, \text{“}IBM''), (price, \leq, 128)\}$, which notifies the investor of the current stock information of IBM when the stock price of the company is not greater than $128.

The event and subscription will match if and only if each constraint in the subscription matching property and corresponding value can be found in the event. It can be seen that for subscription s_1, the event e_1 satisfies the subscription with the values of the attributes *name* and *price*, so e_1 matches s_1.

Threat model

In this scenario, there are three main parties involved: App client, App server, and pub/sub cloud service. App clients and servers are considered trusted in the system while the cloud service is provided by an untrustworthy external infrastructure provider and is in a different management domain from the app. In accordance with the most of pub/sub system privacy protection literature, we apply an “honest but curious” threat model. In this model, while the pub/sub cloud service honestly handles event matching and routing, it also has an interest in learning about the content of events and subscriptions.

Efficient confidentiality protection schemes

In this section, we mainly introduce the proposed confidentiality protection schemes SBM-I and SBM-II and give the security analysis of the schemes respectively.

SBM-I

Approach overview

Unlike previous cryptographic-based confidentiality protection schemes, SBM-I is based on an index-free OPE algorithm. It guarantees the orderliness of ciphertexts by sacrificing the order information of plaintexts. Since the ciphertext is ordered, the event matching algorithm is similar to the plaintext matching scheme at this time, and there is no need to match with the encrypted subscriptions item by item.

We present the construction of SBM-I with an example of the index-free structure-preserving encryption algorithm proposed by Boldyreva et al. The algorithm is based on the natural relationship between the random order-preserving function (ROPF) and the hypergeometric metric probability distribution. The value space of plaintext is [*inrangeStart*, *inrangeEnd*], the value space of ciphertext is [*outrangeStart*, *outrangeEnd*], and the size of ciphertext space is larger than the size of plaintext space. The encryption scheme associates the plaintext space with the ciphertext space, each plaintext value is eventually mapped to a subinterval in the ciphertext space, and then a value is randomly selected

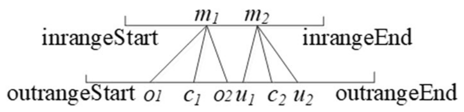

Fig. 3 Correspondence between plaintext and ciphertext

from this subinterval as the corresponding ciphertext value. In Fig. 3, the plaintexts m_1 and m_2 are mapped to subintervals $[o_1, o_2]$ and $[u_1, u_2]$ in the ciphertext space, and then c_1 and c_2 are selected from $[o_1, o_2]$ and $[u_1, u_2]$ as the ciphertexts of m_1 and m_2 respectively. And for m_1, m_2, if $m_1 < m_2$, then $c_1 < c_2$ holds, and vice versa.

In SBM-I, events and subscriptions are encrypted at the app server and then sent to the broker. After receiving the subscription cipher, the broker stores the encrypted subscriptions in an orderly manner by constructing an efficient index structure. When performing the comparison of events and subscriptions, the mismatched subscriptions are filtered out in the index structure, which greatly reduces the subscriptions to be compared. Take the index structure H-tree for example, the algorithmic complexity of both insertion and query is $O(\log n)$.

Security analysis

The confidentiality of the SBM-I scheme is guaranteed by the key (K_{Enc}, K_H), where K_{Enc} is a key for a normal encryption scheme and K_H is a key for an order-preserving hash function H. And the encryption key is known only to the trusted APP server and clients.

In the application scenario studied in this paper, the pub/sub cloud service is considered to be untrustworthy. Attackers can only get information from the broker of the pub/sub cloud service and try to break the system's confidentiality with this information. In the process of subscription insertion and event matching performed by the broker, the only information available to the attacker is the event ciphertext and subscription ciphertext which can reflect the plaintext sequential information. Thus, it is clear that the cryptographic attack on this scheme is a ciphertext-only attack(COA).

Since both event and subscription ciphertexts are encrypted using the OPE algorithm proposed by Boldyreva et al. In their subsequent paper Boldyreva et al. (2011), they state that when the plaintexts are uniformly distributed, the attacker will not obtain the exact value of the plaintexts when encrypting them by appropriately selecting the parameter values, but the attacker can obtain at least half of the plaintext bits. The constructed OPE scheme only satisfies the ROPF security and does not achieve the ideal security of the OPE algorithm IND-OCPA.

SBM-II

Approach overview

The SBM-I algorithm above is less secure due to the leakage of plaintext bits of event and subscription information. In this section, we construct a more secure ciphertext matching algorithm SBM-II based on the MOPE (Popa et al. 2013) algorithm. Although it can guarantee the desired security, the existing MOPE scheme requires frequent interactions during insertion and query between the OPE client and server, which are the App server and the broker at the pub/sub cloud service in our case respectively. This greatly increases the cost of maintaining indexes and queries.

To solve the problem that the broker in the pub/sub system has to interact with the App server frequently to determine the insertion position of ciphers during the insertion and query process, we use the trusted execution environment SGX at the broker (assuming the broker supports Intel's SGX environment). The decryption of ciphertext during insertion and query is performed in the Enclave created by SGX, which eliminates this frequent interaction while achieving the desired security. In addition, SBM-II solves the problem that the original MOPE does not support multi-dimensional conditional queries by adopting balanced binary trees to encode objects in each dimension so that the insertion, encoding, and querying of objects in different dimensions can be performed simultaneously in a multi-threaded manner.

Compared to Pires et al. (2016) and Wang et al. (2021), which perform matching directly inside the Enclave, SBM-II only utilizes SGX to perform decryption operations, which does not lead to frequent page misses or cache misses due to the memory limitation of SGX itself when there is a large-scale subscription in the system. Although the PUBSUB-SGX system proposed by Arnautov et al. (2018) in 2018 evenly divides the subscriptions into individual broker nodes by Load Balancer, which makes the number of stored subscriptions at each broker node decrease and thus reduces the Enclave memory usage, it also doesn't fundamentally solve the problem of SGX memory limitation.

Besides, Intel release the second generation SGX called SGX2 with larger enclave memory to decrease page or cache misses. SGX2 gives software the ability to dynamically add and remove pages from an enclave and to manage the attributes of enclave pages. In our paper, we focus on SGX1 for three reasons. First, SGX1 is the most recent and popular variant of SGX. As a hardware component, SGX1 cannot be easily upgraded to SGX2.

So it's expected that SGX1 will remain the dominant version for a while. Second, our solution achieves unity in efficiency and cost compared to SGX2. The expansion of enclave memory in SGX2 comes with a certain efficiency cost. Initializing and adding memory to the EPC for an enclave is much more complex than a simple memory allocation in a regular application. Third, our SBM-II is designed to work with SGX, programs that use it can operate without any changes in SGX2.

SBM-II consists of five main algorithms, initial state generation, key generation, subscription insertion, subscription encoding, and event matching.

Initial state generation As in Algorithm 1, there are b attributes in the pub/sub system, and empty balanced binary tree T_i is generated at the broker of the pub/sub cloud service for each attribute a_i as the initial state.

Algorithm 1 Initializing State Generation

```
Input: N/A
Output: Balanced Binary Trees $T_1, T_2, \cdots, T_b$
1: for each attribute $a_i$ do
2:     Create balance Tree $T_i$
3: end for
```

Key generation As in Algorithm 2, the symmetric key s_k for AES is generated by the App server, which only shares s_k with the Enclave created at the pub/sub cloud service broker.

Algorithm 2 KeyGen(k)

```
Input: k
Output: sk
1: $sk = AES.KeyGen(l^k)$
2: return sk
```

Subscription insertion As in Algorithm 3, subscription $s_j = \{(a_1, \geq, v_{1j}), (a_2, \geq, v_{2j}), \ldots, (a_b, \geq, v_{bj})\}$. App server encrypts the constraint values corresponding to attribute a_i with sk to get c_{ij}. Then the subscription ciphertext $s'_j = \{(a_1, \geq, c_{1j}), (a_2, \geq, c_{2j}), \ldots, (a_b, \geq, c_{bj})\}$ is sent to the pub/sub cloud service. The broker in the pub/sub cloud service inserts each constraint ciphertext c_{ij} into the tree T_i corresponding to different attributes a_i, where r_i is the root node of tree T_i and n_j is the node with the same value of c_{ij} when c_{ij} in tree T_i.

Algorithm 3 Insert to subscription

```
Input: $s'_j, \{T_1, T_2, ....T_b\}$
Output: N/A
1: for attribute $a_i$ do
2:     if $isExit(c_{ij}, T_i) == False$ then
3:         $compare(r_i, c_{ij})$
4:         if $r_i! = null$ then
5:             $u \leftarrow CompareAtEnclave(c_{ri}, c_{ij})$
6:             if $u == -1$ then
7:                 compare($r_i.rightChild, c_{ij}$)
8:             else
9:                 compare($r_i.leftChild, c_{ij}$)
10:            end if
11:        else
12:            insertValue($c_{ij}, T_i$)
13:            Encode($T_i$)
14:        end if
15:    else
16:        $n_j$.addID($id_j$)
17:    end if
18: end for
```

Subscription encoding As in Algorithm 4, subscription encoding is building indexes for subscriptions. To ensure that the improved OPE scheme can achieve the desired security, the tree T_i needs to update the encoding at the broker after each insertion of constraint values. In order to support the matching of equal-value, non-equal-value and interval-type subscriptions in the pub/sub system, a new encoding method is reconstructed in this paper.

Algorithm 4 Subscription encoding algorithm

```
Input: $T_i$
Output: N/A
1: $d \leftarrow depth(T_i)$, $h = d - 2(d \geq 2)$
2: $r_i$.encode $= 2^0 + 2^i + \ldots + 2^h$
3: if $r_i$.leftChild != null then
4:     $r_i$.leftChild.encode = $r_i$.encode-$2^h$
5:     computeEncode($r_i$.leftChild, h-1)
6: end if
7: if $r_i$.rightChild != null then
8:     $r_i$.rightChild.encode = $r_i$.encode + $2^h$
9:     computeEncode($r_i$.rightChild, h-1)
10: end if
```

Event matching As in Algorithm 5, the event $e_k = \{(a_1, v_{1k}), (a_2, v_{2k}), \ldots, (a_b, v_{bk})\}$, each attribute value v_{ik} is encrypted with sk at the App server to get c_{ik}, and then the event ciphertext $e'_k = \{(a_1, c_{1k}), (a_2, c_{2k}), \ldots, (a_b, c_{bk})\}$ is sent to the pub/sub cloud service. The broker in the pub/sub cloud service performs event matching and finds the list of subscription IDs matching the event.

Algorithm 5 Event matching

```
Input: $e'_k$, $\{T_1, T_2, \ldots, Tb\}$
Output: Id-list, corresponding subscription matching event $e_k$
 1: for $a_i$ do
 2:     if isExist($s_{ik}$, $T_i$)==false then
 3:         insert($c_{ik}$, $T_i$)
 4:     end if
 5:     $e_{ik} \leftarrow$ getEncode($T_i$, $c_{ik}$)
 6:     $Id\text{-}list_1 \leftarrow$ getMatchId($e_{ik}, T_i$)
 7:     if $Id\text{-}list_1$ != null then
 8:         Id-list = Id-list $\wedge$ $Id\text{-}list_1$
 9:     else
10:         return null
11:     end if
12: end for
13: return Id-list
```

Implement of SBM-II

We give examples of subscription insertion and event matching to describe our scheme.

Table 1 gives the subscription information in the system and the process of insertion is as follows.

(1) The subscription information sent by the user is encrypted at the APP server and then sent to the broker in the pub/sub cloud service.
(2) Take subscription s_1 as an example, first find the corresponding OPE trees T_1 and T_2 for attributes a_1 and a_2 in subscription s_1. Take attribute a_1 and tree T_1 as an example, then determine whether the constraint ciphertext on attribute a_1 exists in tree T_1. If it does, directly add the subscription ID of s_1 to the Id chain table of the equivalent node, and the insertion process is finished. If it does not exist, the constraint ciphertext corresponding to attribute a_1 needs to be ciphertext is inserted into the tree.
(3) If the tree T_1 is empty, the constraint cipher on a_1 is directly inserted into the root node of tree T_1, and the subscription ID is stored in the Id chain table of the root node.
(4) If the tree T_1 is not empty at the time of insertion, the ciphertext at the root node is sent to the Enclave created by the broker along with the ciphertext of the node to be inserted if the constraint values of these subscriptions on attribute a_1 do not exist in tree T_1. Conversely, if the constraint value of attribute a_1 exists in tree T_1, then it is processed as described in (2).
(5) Decrypt the two ciphertexts in the Enclave, compare the corresponding plaintext sizes, and return the comparison result.
(6) If the value to be inserted is smaller than that of the root node, the left child node of the root is selected and compared with the ciphertext value to be inserted. If the left child node is empty, the ciphertext is directly inserted into the left child node, and the subscription ID is stored in the ID chain table of the left child node. Otherwise, the left child node is used as the root node, and the execution starts back to (4).
(7) If the cipher to be inserted is larger than the cipher value of the root node, the right child node of the root is selected and compared with the cipher value to be inserted. If the right child node is empty, the cipher is directly inserted into the right child node, and the subscription ID is stored in the Id chain table of the right child node. Otherwise, the right child node is used as the root node, and the execution starts back to (4).
(8) When the insertion of the constraint ciphertext on attribute a_1 is completed, the tree T_1 is rebalanced and the encoding algorithm is invoked to re-encode the tree to generate the order-preserving ciphertext.

Table 1 The Subscriptions in our system

Sub-list	ID	Sub
s_1	1	$[(a_1, \geq, 10) \wedge (a_2, \geq, 19)]$
s_2	2	$[(a_1, \geq, 13) \wedge (a_2, \geq, 55)]$
s_3	3	$[(a_1, \geq, 10) \wedge (a_2, \geq, 85)]$
s_4	4	$[(a_1, \geq, 23) \wedge (a_2, \geq, 19)]$
s_5	5	$[(a_1, \geq, 17) \wedge (a_2, \geq, 32)]$
s_6	6	$[(a_1, \geq, 75) \wedge (a_2, \geq, 22)]$

Figure 4 shows the state of the tree after the insertion of $s_1 - s_6$. In the OPE tree T_1 corresponding to attribute a_1, the encoded values of each constraint cipher (order preserving cipher) are $\{1, 3, 1, 5, 4, 6\}$, and in the OPE tree T_2 corresponding to attribute a_2, the encoded values of each constraint cipher are $\{0, 3, 5, 0, 2, 1\}$. The OPE tree corresponding to an attribute must be rebalanced each time a value for that attribute is inserted. Therefore, the generated order-preserving ciphertext changes with the position of the node, and the variability of the OPE ciphertext is a necessary condition for the OPE algorithm to achieve the desired security.

This case gives the case when the subscription constraint is non-equal ($\geq$). When constructing an OPE tree, the number of trees constructed depends not only on the number of attributes but also on the type of constraints. For example, if all the constraints in the system are of interval type, then for each attribute, the system will build two trees to store the ciphertext at the low and high values of the attribute set by

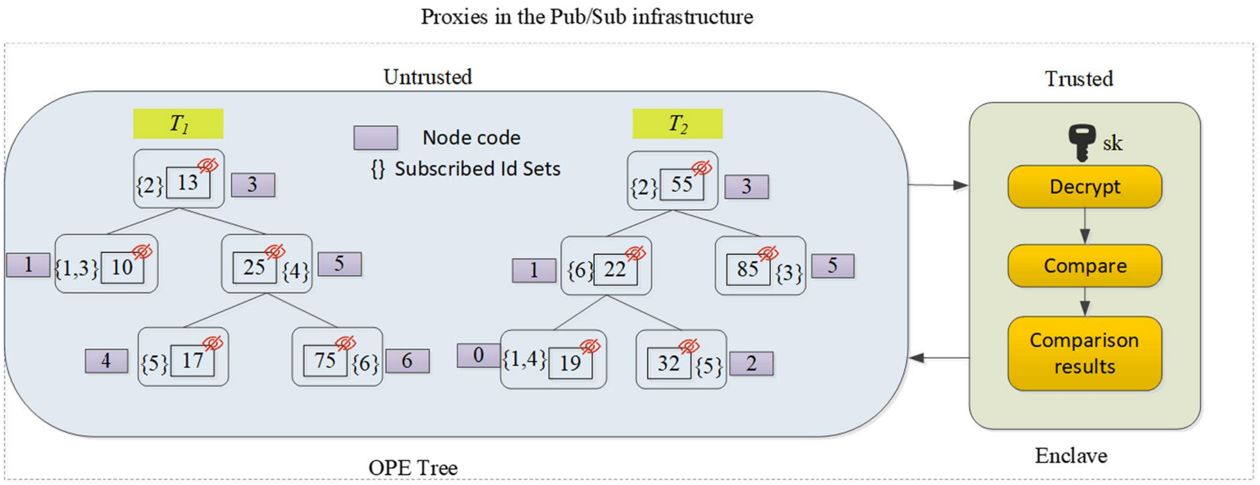

Fig. 4 The state of OPE tree after inserting subscriptions

each subscription. When performing matching, for a subscription, if the constraint values in both trees for each attribute satisfy the attribute values of the event on the same attribute, then it matches the event; otherwise, it does not match.

The event information in the system is given in Table 2, and the event matching process is as follows:

(1) Events published by publishers are encrypted at the App server and sent to the broker in the pub/sub cloud service.
(2) For the attributes a_1, a_2 in the event, find the corresponding OPE trees T_1 and T_2, respectively, and determine whether the attribute value on a_1 exists in tree T_1 and the attribute value on a_2 exists in tree T_2. If it exists, get the OPE code of the equivalent node. For example, if the value of event e_1 on attribute a_2 exists in the tree, return code 6.
(3) If it does not exist, insert the event ciphertext into the tree according to Algorithm 3, and then rebalance the tree to find its order-preserving encoding. For example, the attribute values of event e_1 and e_2 on attribute a_1, and event e_2 on a_2 will be inserted into T_1 and T_2 respectively.
(4) When matching, find the subscription nodes in T_1 and T_2 corresponding to codes less than or equal to the event mapped to on a_1 and a_2, respectively, and take the intersection of the ID chains corresponding to these nodes. The remaining set of IDs is the ID corresponding to the subscription matching the event.

Figure 5 gives the state of the trees T_1 and T_2 when matching is performed. It can be seen that for e_1, the subscription IDs satisfied on attribute a_1 are $\{1, 2, 3, 5\}$ and on attribute a_2 are $\{1, 4\}$. The corresponding codes of these subscriptions in the tree are less than or equal to the codes of the event on the same attribute, and the intersection of the two is taken to be $\{1\}$, which is the matching subscription ID. Similarly, the set of subscription IDs for matching e_2 is obtained as $\{1, 4, 5\}$.

Security analysis

Before subscription insertion, there is only an empty OPE tree, since the attacker does not have the AES encryption key, the original subscription plaintext cannot be recovered from the subscription ciphertext encrypted using AES symmetric encryption algorithm.

In the subscription insertion process in Algorithm 3, the decryption of the subscription is performed under the trusted execution environment SGX, and the attacker does not get the subscription plaintext. After the subscription insertion is completed, the encoding of the subscriptions in the tree can be obtained from the generated OPE tree, and the encoding only exposes the order relationship between the subscriptions.

When performing event matching, the event code is first inserted into the tree to get the event code. Then we search for the node with the matching event code. Similar to the subscription insertion, the event insertion and the process of getting the matching node only compare the codes and do not expose the event ciphertext.

Since the attacker does not have access to the AES symmetric key, the SBM-II implementation exposes the event and subscription encoding, but not the event and subscription plaintext information. In addition, the attacker can get no information from the encoding other than the order of events and subscriptions.

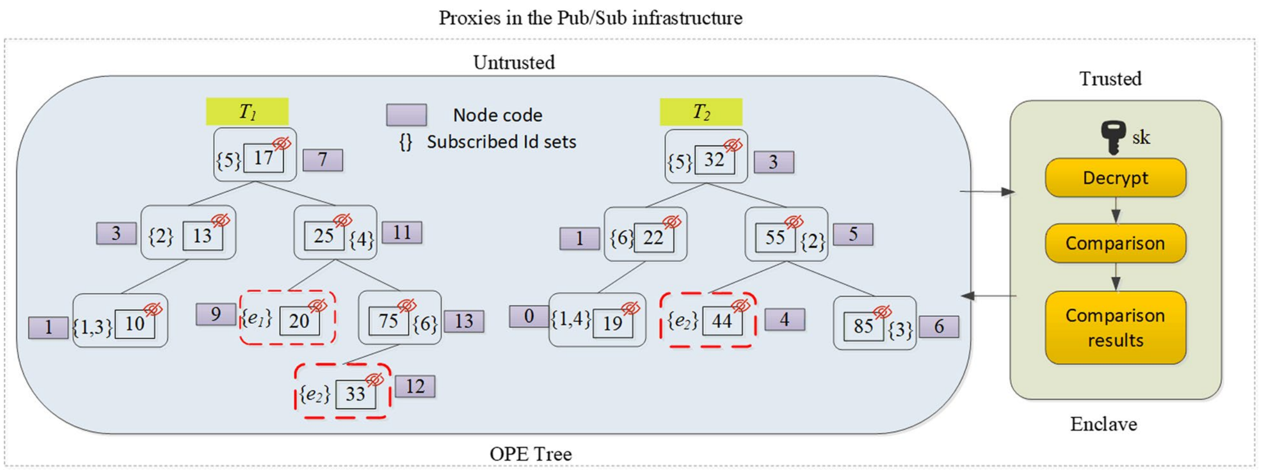

Fig. 5 The state of OPE tree when matching events

Evaluation

Workload

During the experiment, the broker is running on a Win10 host with Intel(R) Core(TM) i5-1035G1 CPU and 16GB RAM, and the APP server is running on a Win10 host with Intel(R) Core(TM) i5-6200U CPU and 8 GB RAM. SGX is developed using the SDK provided by Intel.

The number of events and subscriptions are uniformly random integers generated on [0,1000], and the subscriptions generated in the experiment are all interval-type subscriptions, the number of events is 1000 when matching is performed, the number of event attributes and subscription constraints is 4, and the size of the subscription set is between 100,000 and 500,000. In SBM-I, the security length of the encryption key is chosen to be 256bits when encrypting events and subscriptions using the OPE algorithm proposed by Boldyreva et al. We use the tree index structure of H-Tree (Qian et al. 2014) as a representative to verify our algorithm.

Approach

Event matching time is the most important metric for matching algorithms. Matching time is defined as the time taken by a broker to find a matching set of subscriptions when an event arrives. In our experiments, we measure the average matching time for a single event.

To measure the speed of the proposed matching algorithm by indexing subscriptions and the existing ciphertext matching algorithms, we select one from each of the cryptography-based and SGX-based algorithms as comparisons. The matching algorithm based on Paillier homomorphic encryption(MP) proposed by Nabeel et al. (2013) and the SGX-based SCBR algorithm proposed by Pires et al. (2016) are chosen, and the length of the security key is set to 1024 bits. We measured the impact of the number of subscriptions on the average matching time using SBM-I, SBM-II, MP, and SCBR.

Furthermore, to assess the impact of frequent interactions on the matching performance of SBM-II, we analyzed the average matching time when the Enclave container is generated at the broker and when it is not produced respectively. The broker must communicate with the APP server if an Enclave container has not yet been formed.

Table 2 The events in our system

Pub-list	Pub
e_1	$[(a_1, 20), (a_2, 19)]$
e_2	$[(a_1, 33), (a_2, 44)]$

Results and analysis

Impact of the matching algorithm

Figure 6 shows the variation of event matching time with the increasing number of subscriptions, and the vertical coordinate displays the result of the operation after taking log 10. Although the average matching time of four algorithms increases with the number of subscriptions, the matching performance of SBM-I and SBM-II is significantly better than MP and SCBR. When the number of subscriptions is 500,000, SCBR matches roughly twice as fast as MP, the matching time of SBM-I is 3–4 orders of magnitude faster than MP and SCBR, and the matching time of SBM-II is 24 times faster than SCBR and 40 times faster than MP.

The reason is that the use of the index structure directly filters out the mismatched subscriptions, which greatly reduces the number of subscriptions to be matched. In cryptographic algorithm-based ciphertext

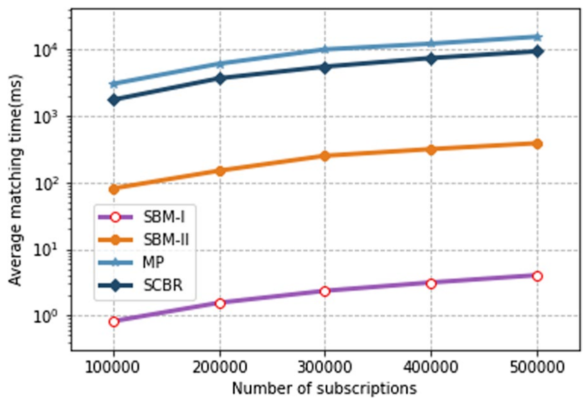

Fig. 6 The effect of the number of subscriptions on the average matching time

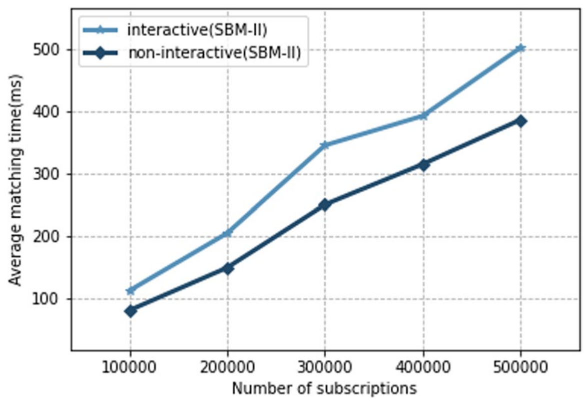

Fig. 7 The effect of elimination of frequent interactions on the average matching time

matching schemes, events need to be compared with the subscriptions one by one, so its performance is lower than that of the SBM scheme. In SGX-based schemes, the increase in the number of subscriptions causes frequent page transfers in and out, and the system disk will run out of resources. Additionally, it demonstrates that while the security of SBM-II is higher than that of SBM-I its efficiency is lower than that of SBM-I.

Effect of elimination of frequent interactions

To measure the improvement of SBM-II over the original MOPE, we measure the average matching time with and without interaction between the APP server and the broker in the published subscription cloud service when performing matching as the number of subscriptions increases. The two hosts communicate in the LAN when there is interaction.

As can be seen from Fig. 7, when the number of subscriptions is 500,000 and there is no interaction, SBM-II improves the average matching time by 23% over that with interaction. It verifies that the elimination of frequent interactions reduces the overhead of the system when inserting subscriptions and performing matching, and improves the performance of the system in general.

Conclusion

In this paper, we propose an efficient confidentiality protection scheme SBM for a content-oriented pub/sub system. SBM uses OPE algorithms to ensure the confidentiality of the system while achieving orderly storage of cipher text and constructs indexes for subscription ciphertexts to achieve efficient matching of large-scale subscriptions. SGX is applied to the algorithm SBM-II to improve the performance of encrypted event matching by indexing subscription ciphertexts, and eventually meet the need for efficient matching of large-scale pub/sub system. The experiment results show that the indexing of subscriptions has better performance than the traditional cryptography-based matching methods. While SBM-I is more efficient than SBM-II matching, SBM-II is more secure. There are still areas that can be improved for SBM-II, though the scheme SBM-II can achieve the desired security of the OPE algorithm, solve the original interaction problem, and reduce the cost of insertion and query. For example, we do not consider the handling of equivalence speculation attacks and frequency analysis attacks in our scheme, which can be further explored in later studies.

Acknowledgements
We thank Prof. Juan Wang and Dr. Jie Wang of Wuhan University for their guidance and support in the SGX experiment.

Author Contributions
All authors have contributed to this manuscript and approve of this submission. JP, YS and QF participated in the experiment and drafting the article. Prof. RS has made a decisive contribution to the technical route, designing research, and revising the article critically. LL, SY, JS, and ZM have made many contributions on revising the article. All authors read and approved the final manuscript.

Funding
This work was supported by the Natural Science Foundation of Beijing Municipality (M21037), Key Technologies Research and Development Program (2022YFF0902701), 2022 Industrial Internet Public Service Platform - Industrial Internet Oriented Virtual Currency Mining Governance Public Service Platform Project by the Ministry of Industry and Information Technology of PRC, Major Research and Application Project for the Supervision Platform of Virtual Currency Mining Behavior by the Ministry of Education of PRC, and the 111 Project (Grant No. B21049).

Availability of data and materials
The authors confirm that the data supporting the findings of this study are available within the article.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 11 March 2023 Accepted: 6 June 2023
Published online: 04 July 2023

References

Agrawal R, Kiernan J, Srikant R, Xu Y (2004) Order preserving encryption for numeric data. In: Proceedings of the 2004 ACM SIGMOD international conference on management of data, pp 563–574

Amazon (2022) Pub/Sub messaging. https://aws.amazon.com/pub-sub-messaging

Arnautov S, Brito A, Felber P, Fetzer C, Gregor F, Krahn R, Ozga W, Martin A, Schiavoni V, Silva F et al (2018) Pubsub-sgx: exploiting trusted execution environments for privacy-preserving publish/subscribe systems. In: 2018 IEEE 37th symposium on reliable distributed systems (SRDS), pp 123–132. IEEE

Barazzutti R, Felber P, Mercier H, Onica E, Riviere E (2015) Efficient and confidentiality-preserving content-based publish/subscribe with prefiltering. IEEE Trans Dependable Secure Comput 14(3):308–325

Bethencourt J, Sahai A, Waters B (2007) Ciphertext-policy attribute-based encryption. In: 2007 IEEE symposium on security and privacy (SP'07), pp 321–334 . IEEE

Bloom BH (1970) Space/time trade-offs in hash coding with allowable errors. Commun ACM 13(7):422–426

Boldyreva A, Chenette N, Lee Y, O'neill A (2009) Order-preserving symmetric encryption. In: Annual international conference on the theory and applications of cryptographic techniques, pp 224–241. Springer

Boldyreva A, Chenette N, O'Neill A (2011) Order-preserving encryption revisited: improved security analysis and alternative solutions. In: Annual cryptology conference, pp 578–595 (2011). Springer

Borcea C, Polyakov Y, Rohloff K, Ryan G et al (2017) Picador: end-to-end encrypted publish-subscribe information distribution with proxy re-encryption. Future Gener Comput Syst 71:177–191

Carzaniga A, Rosenblum DS, Wolf AL (2001) Design and evaluation of a wide-area event notification service. ACM Trans Comput Syst (TOCS) 19(3):332–383

Choi S, Ghinita G, Bertino E (2010) A privacy-enhancing content-based publish/subscribe system using scalar product preserving transformations. In: International conference on database and expert systems applications, pp 368–384. Springer, Berlin

Ding T, Qian S, Cao J, Xue G, Li M (2020) Scsl: optimizing matching algorithms to improve real-time for content-based pub/sub systems. In: 2020 IEEE international parallel and distributed processing symposium (IPDPS), pp 148–157. IEEE

Dong C, Russello G, Dulay N (2011) Shared and searchable encrypted data for untrusted servers. J Comput Secur 19(3):367–397

Eugster PT, Felber PA, Guerraoui R, Kermarrec A-M (2003) The many faces of publish/subscribe. ACM Comput Surv (CSUR) 35(2):114–131

Gaballah SA, Coijanovic C, Strufe T, Mühlhäuser M (2021) 2PPS—publish/subscribe with provable privacy. In: 2021 40th international symposium on reliable distributed systems (SRDS), pp 198–209. IEEE

Google (2022) Pubsub. https://cloud.google.com/pubsub/docs/overview

Guo J, Miao M, Wang J (2018) Research and progress of order preserving encryption. J Cryptol Res 5:182–195

Ion M, Russello G, Crispo B (2012) Design and implementation of a confidentiality and access control solution for publish/subscribe systems. Comput Netw 56(7):2014–2037

Ji S, Jacobsen H-A (2018) Ps-tree-based efficient Boolean expression matching for high-dimensional and dense workloads. Proc VLDB Endow 12(3):251–264

Kumar S, Hu Y, Andersen MP, Popa RA, Culler DE (2019) {JEDI}: {Many-to-Many} {End-to-End} encryption and key delegation for {IoT}. In: 28th USENIX security symposium (USENIX Security 19), pp 1519–1536

Liang J, Qin Z, Xiao S, Zhang J, Yin H, Li K (2020) Privacy-preserving range query over multi-source electronic health records in public clouds. J Parallel Distrib Comput 135:127–139

Microsoft (2022) Publisher-subscriber pattern. https://learn.microsoft.com/zh-cn/azure/architecture/patterns/publisher-subscriber

Nabeel M, Appel S, Bertino E, Buchmann A (2013) Privacy preserving context aware publish subscribe systems. In: International conference on network and system security, pp 465–478. Springer, Berlin

Paillier P (1999) Public-key cryptosystems based on composite degree residuosity classes. In: International conference on the theory and applications of cryptographic techniques, pp 223–238. Springer

Pal P, Lauer G, Khoury J, Hoff N, Loyall J (2012) P3s: a privacy preserving publish-subscribe middleware. In: ACM/IFIP/USENIX international conference on distributed systems platforms and open distributed processing, pp 476–495. Springer, Berlin

Pires R, Pasin M, Felber P, Fetzer C (2016) Secure content-based routing using intel software guard extensions. In: Proceedings of the 17th international middleware conference, pp 1–10

Popa RA, Li FH, Zeldovich N (2013) An ideal-security protocol for order-preserving encoding. In: 2013 IEEE symposium on security and privacy, pp 463–477. IEEE

Qian S, Cao J, Zhu Y, Li M, Wang J (2014) H-tree: an efficient index structure for event matching in content-based publish/subscribe systems. IEEE Trans Parallel Distrib Syst 26(6):1622–1632

Qian S, Cao J, Zhu Y, Li M (2014) Rein: a fast event matching approach for content-based publish/subscribe systems. In: IEEE INFOCOM 2014-IEEE conference on computer communications, pp 2058–2066. IEEE

Raiciu C, Rosenblum DS (2006) Enabling confidentiality in content-based publish/subscribe infrastructures. In: 2006 securecomm and workshops, pp 1–11. IEEE

Shikfa A, Önen M, Molva R (2009) Privacy-preserving content-based publish/subscribe networks. In: IFIP international information security conference, pp 270–282. Springer, Berlin

Tariq MA, Koldehofe B, Rothermel K (2013) Securing broker-less publish/subscribe systems using identity-based encryption. IEEE Trans Parallel Distrib Syst 25(2):518–528

Wang C, Carzaniga A, Evans D, Wolf AL (2002) Security issues and requirements for internet-scale publish-subscribe systems. In: Proceedings of the 35th annual hawaii international conference on system sciences, pp 3940–3947. IEEE

Wang S, Pan D, Feng R, Zhang Y (2021) Magikcube: securing cross-domain publish/subscribe systems with enclave. In: 2021 IEEE 20th international conference on trust, security and privacy in computing and communications (TrustCom), pp 147–154. IEEE

Wong WK, Cheung DW-l, Kao B, Mamoulis N (2009) Secure kNN computation on encrypted databases. In: Proceedings of the 2009 ACM SIGMOD international conference on management of data, pp 139–152

Publisher's Note