Hindawi
Security and Communication Networks
Volume 2017, Article ID 4215425, 15 pages
https://doi.org/10.1155/2017/4215425

Research Article

Authenticated Location-Aware Publish/Subscribe Services in Untrusted Outsourced Environments

Han Yan, Xiang Cheng, Sen Su, and Siyao Zhang

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China

Correspondence should be addressed to Sen Su; susen@bupt.edu.cn

Received 7 March 2017; Revised 10 June 2017; Accepted 21 June 2017; Published 25 September 2017

Academic Editor: Jiankun Hu

Copyright © 2017 Han Yan et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Location-aware publish/subscribe is an important location-based service based on server-initiated model. Often times, the owner of massive spatio-textual messages and subscriptions outsources its location-aware publish/subscribe services to a third-party service provider, for example, cloud service provider, who is responsible for delivering messages to their relevant subscribers. The issue arising here is that the messages delivered by the service provider might be tailored for profit purposes, intentionally or not. Therefore, it is essential to develop mechanisms which allow subscribers to verify the correctness of the messages delivered by the service provider. In this paper, we study the problem of authenticating messages in outsourced location-aware publish/subscribe services. We propose an authenticated framework which not only can deliver the messages efficiently but also can make the subscribers' authentication available with low cost. Extensive experiments on a real-world dataset demonstrate the effectiveness and efficiency of our proposed authenticated framework.

1. Introduction

With the rapid development of mobile Internet and positioning-enabled devices (e.g., smart phones), massive amount of data that contain both text information and geographical location information are being generated at an unprecedented scale on the Web. This enables location-based services ($\mathscr{LBS}$), such as Foursquare (https://foursquare.com) and Yelp (https://www.yelp.com), to be extensively deployed in many systems and widely accepted by Internet users. Location-aware publish/subscribe is an important kind of service based on server-initiated model (relative to user-initiated model, like spatial-keyword query) in $\mathscr{LBS}$. For example, in a Groupon system, subscribers register their spatio-textual subscriptions to capture their interests (e.g., "Adidas shoe discount at Beijing, China") (for the rest of this paper, we use "subscriber" and "subscription" interchangeably if the context is clear). For each Groupon message with textual description and location (e.g., "Adidas running shoes at cheap prices at Adidas factory store, Beijing, China"), the system delivers the message to relevant subscribers.

Since location-aware publish/subscribe is a compute-intensive task, if the data owner of massive spatio-textual messages and subscriptions wants to efficiently deliver each message to relevant subscribers, to strengthen its ability of computing, it needs to build up basic IT infrastructure and hire specialized personnel. However, as such cost might be unaffordable for small-to-medium businesses, outsourcing the data and computations to a third-party service provider (e.g., a cloud service provider) has been an appealing option. Yet, this outsourcing model presents a great challenge that the messages delivered by the service provider might be incomplete or incorrect. There are a variety of reasons for this. First, the service provider might deliver tailored messages to favor its sponsors. Second, the service provider might use some inferior algorithms and deliver the suboptimal messages to the subscribers to save computing resources. Third, with the growing popularity of the cloud, more and more security breaches and attacks on such systems have been reported. In case an attacker takes control of the service provider's server, it may forge the messages for its own interest.

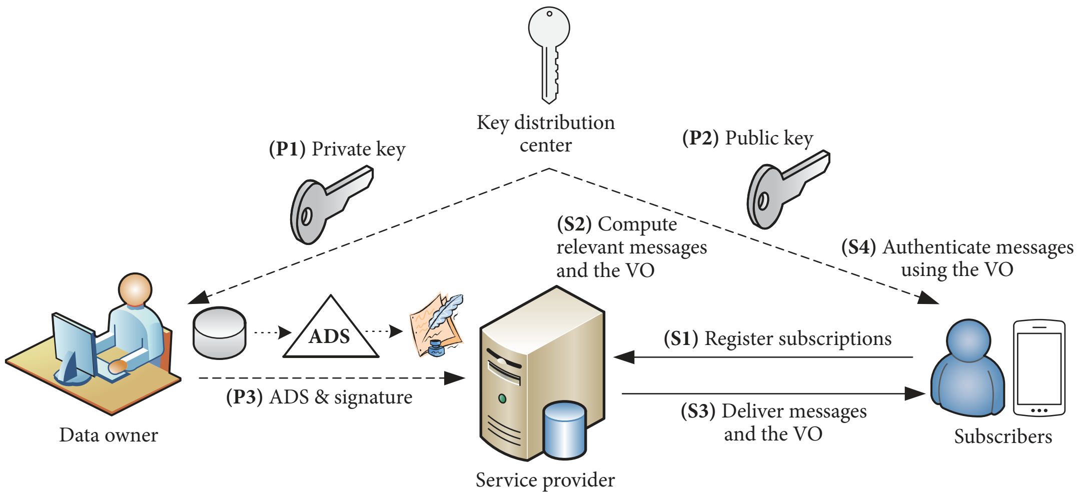

Key distribution center
(P1) Private key
(P2) Public key
(S2) Compute relevant messages and the VO
(S4) Authenticate messages using the VO
ADS
(S1) Register subscriptions
(P3) ADS & signature
(S3) Deliver messages and the VO
Data owner
Service provider
Subscribers

FIGURE 1: System model.

The aforementioned reasons necessitate the development of mechanisms that allow subscribers to authenticate the messages delivered by the service provider. They should be verified in terms of two conditions: (1) *soundness* and (2) *completeness*. The former means that the messages are not tampered with, while the latter implies that no valid message is missing.

In this paper, to make one step further towards practical deployment of location-aware publish/subscribe in untrusted outsourcing environments, we study the problem of authenticating messages in outsourced location-aware publish/subscribe services. To address this problem, we present an authenticated location-aware publish/subscribe framework. We assume that messages are allowed a maximum delay Δt to be delivered to their corresponding subscribers. The data owner organizes the messages within $\Delta t'$ ($\Delta t' \leq \Delta t$) in an authenticated data structure ($\mathscr{ADS}$) called TMR-tree. Based on the TMR-tree, the service provider first computes the relevant messages for each subscription. During this process, we present an inverted index pruning technique to reduce the times of inverted index (used to index the subscriptions) traversal, thus improving the efficiency of computing the relevant messages for each subscription. Then, the service provider constructs a verification object ($\mathscr{VO}$) for each subscription and the corresponding subscriber can authenticate the messages delivered to it. A thorough experimental study on a real-world dataset is conducted over a wide range of workload settings to evaluate the effectiveness and efficiency of our proposed framework in terms of various performance metrics.

Roadmap. The rest of this paper is organized as follows. Section 2 introduces some preliminaries, which include system model, problem definition, and background knowledge. Section 3 presents our proposed authenticated location-aware publish/subscribe framework. In Section 4, we experimentally evaluate the performance of our proposed framework. Related work on the location-aware publish/subscribe and authenticated query processing is surveyed in Section 5. In the end, we conclude the paper in Section 6.

2. Preliminaries

In this section, we first describe our system model. Then, we define the problem studied in this paper. At last, we introduce some background knowledge on cryptographic primitives and location-aware publish/subscribe which underlie our proposed framework.

2.1. System Model. As shown in Figure 1, our system involves four entities: the data owner, the service provider, the subscribers, and the key distribution center ($\mathscr{KDC}$).

First, the data owner builds an authenticated data structure ($\mathscr{ADS}$) over the messages within $\Delta t'$ ($\Delta t' \leq \Delta t$; recall that Δt is a predefined maximum permissible delivery delay) and signs the $\mathscr{ADS}$ using the private key distributed by the $\mathscr{KDC}$. Then, the data owner outsources the location-aware publish/subscribe services to the service provider, who provides the storage resources for the messages, the $\mathscr{ADS}$, the signature of the $\mathscr{ADS}$, and algorithms. Based on the $\mathscr{ADS}$, the service provider finds the messages which are relevant to the registered subscriptions and constructs a verification object ($\mathscr{VO}$) for each subscription. After that, the service provider delivers the messages and the $\mathscr{VO}$ to corresponding subscribers. The subscribers authenticate the *soundness* and *completeness* of these messages using the $\mathscr{VO}$ and the public key distributed by the $\mathscr{KDC}$.

Throughout this paper, we assume that (1) the $\mathscr{KDC}$ and the data owner are trusted but the service provider is the potential adversary and might fabricate the messages (intentionally or not); (2) the $\mathscr{KDC}$ or the data owner does not collude with the service provider; (3) the computation and storage capacities of the service provider are polynomially bounded.

2.2. Problem Definition. In this paper, we study the problem of authenticating messages in outsourced location-aware publish/subscribe services. That is, the subscribers register their interests as subscriptions in the system first. Then, the service provider not only needs to efficiently deliver the messages within $\Delta t'$ to the relevant subscribers whose subscriptions have high relevancy to the messages, but also needs to construct a $\mathscr{VO}$ for each subscriber to allow them to authenticate the *soundness* and *completeness* of the delivered messages. The $\mathscr{VO}$ should be constructed as small as possible for minimizing the communication cost between the service provider and subscribers. Meanwhile, the $\mathscr{VO}$ should be suitable for subscribers' authentication for minimizing the computational cost at the subscribers side.

2.3. Background Knowledge

2.3.1. Cryptographic Primitives. We present the essential cryptographic primitives on one-way hash function, cryptographic signature, and Merkle hash tree as follows.

One-Way Hash Function. A one-way hash function $h(\cdot)$ maps a message m of arbitrary length to a fixed-length output $\mathscr{H}(m)$. It works in one direction. It is easy to compute $\mathscr{H}(m)$ for a message m. However, it is computationally infeasible to find a message that maps to a given $\mathscr{H}(\cdot)$.

Cryptographic Signature. A cryptographic signature (or simply signature) is a mathematical scheme for demonstrating the authenticity of a digital message. A signer applies for a pair of private key and public key from the $\mathscr{KDC}$. The former is kept by the signer secretly and the latter is publicly distributed. A digital message can be signed using the private key. The authenticity of the message can be verified by anyone who receives this message using the public key.

Merkle Hash Tree. The Merkle hash tree ($\mathscr{MHT}$) [1] is an authenticated data structure used for collectively authenticating a set of messages. The $\mathscr{MHT}$ is a binary tree and built in a bottom-up manner, by first computing the hash values of the messages in leaf nodes. The hash value of each internal node is derived from its two children nodes. Finally, the hash value of the root is signed by the owner of the messages. The $\mathscr{MHT}$ can be used to authenticate any subset of messages, in conjunction with a *proof*. The *proof* consists of the signed root and sibling nodes (auxiliary hash values) on the path from the root down to the messages which need to be authenticated.

2.3.2. Location-Aware Publish/Subscribe. We present the state-of-the-art method [2] for location-aware publish/subscribe as follows.

A location-aware publish/subscribe service delivers each message, denoted by $m = (m.loc, m.text)$, to its relevant subscribers who register spatio-textual subscriptions (each subscription is denoted by $s = (s.loc, s.text)$) to capture their interests. $m.loc$ ($s.loc$) is a spatial location with the latitude and longitude. $m.text$ ($s.text$) is a set of keywords $\{t_1, t_2, \dots, t_{|m.text|}\}$ ($\{t_1, t_2, \dots, t_{|s.text|}\}$) and each keyword is associated with a weight $w(t_i)$ which can be set as the inverted document frequency ($\mathscr{IDF}$) of the keyword. To quantify the relevancy between a subscription and a message, [2] used a spatio-textual similarity function

$$\mathbf{SIM}(s,m) = \delta \cdot \mathbf{TSIM}(s,m) + (1-\delta) \cdot \mathbf{SSIM}(s,m), \quad (1)$$

where

$$\mathbf{TSIM}(s,m) = \frac{\sum_{t \in s.text \cap m.text} w(t)}{\sum_{t \in s.text} w(t)} \quad (2)$$

is a textual similarity function which is similar to the weighted Jaccard coefficient and

$$\mathbf{SSIM}(s,m) = \max\left(0, 1 - \frac{\mathbf{DIST}(s.loc, m.loc)}{\mathbf{maxDIST}}\right) \quad (3)$$

is a spatial similarity function, where $\mathbf{DIST}(s.loc, m.loc)$ is the Euclidian distance between $s.loc$ and $m.loc$, and $\mathbf{maxDIST}$ is the maximum user-tolerated Euclidian distance between subscriptions and messages (which can be set as the maximum distance between subscriptions). δ is a preference parameter to tune the weight of textual and spatial similarity. A subscription s and a message m are called relevant if their similarity exceeds a threshold τ. Since subscribers usually have different preferences and requirements on δ and τ (e.g., some subscribers prefer highly relevant results while some subscribers want to get more results), subscribers are allowed to parameterize their parameters δ and τ. Therefore, a parameterized spatio-textual subscription can be redefined as $s = (s.loc, s.text, s.\delta, s.\tau)$. Figure 2 shows an example of 11 parameterized spatio-textual subscriptions and 7 messages.

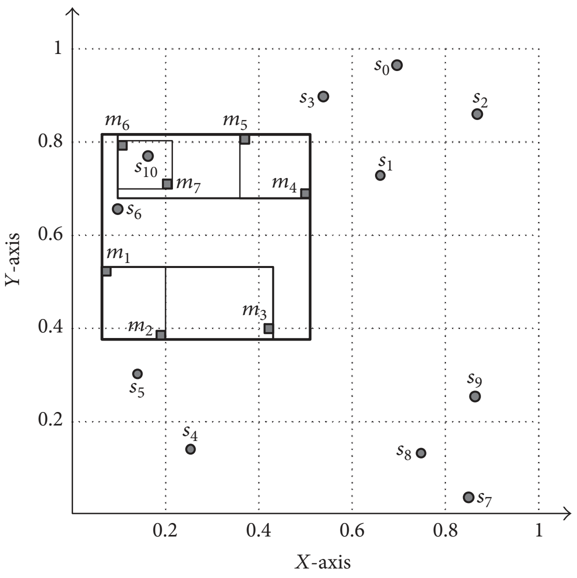
To deliver messages to relevant subscribers efficiently, [2] proposed a *spatial-oriented prefix* to prune irrelevant subscriptions and devised a filter-verification framework. In particular, with respect to the textual filter, [2] claimed that if a subscription s is relevant to a message m, they must share at least one common keyword in the so-called *prefix* of s, which is computed from the textual similarity threshold. More specifically, based on (1), given a subscription s, since the spatial similarity cannot exceed 1, [2] deduced a textual similarity threshold

$$s.\tau^{\mathrm{T}} = \frac{s.\tau - (1 - s.\delta)}{s.\delta}. \quad (4)$$

When $s.\tau^{\mathrm{T}} > 0$, based on $s.\tau^{\mathrm{T}}$, [2] selected a *prefix* for each subscription s. The keywords in $s.text$ are sorted by their weights in descending order and a minimum p such that

$$\frac{\sum_{i=p}^{|s.text|} w(t_i)}{w(s.text)} < s.\tau^{\mathrm{T}} \quad (5)$$

is computed, where $w(s.text)$ is the total weight of keywords in $s.text$. Therefore, the *prefix* of s can be defined as $\mathscr{SIG}(s) = \{t_1, t_2, \dots, t_{p-1}\}$. Since the total weight of keywords after t_p is smaller than $s.\tau^{\mathrm{T}}$, if a subscription s is relevant to a message m (i.e., $\mathbf{TSIM}(s,m) \ge s.\tau^{\mathrm{T}}$), they must share at least one common keyword in $\mathscr{SIG}(s)$.



subscriptions (s.loc, s.text, s.δ, s.τ)	*messages (m.loc, m.text)*
s_0: $((0.70, 0.96), \{t_4, t_2\}, 0.7, 0.8)$	m_1: $((0.07, 0.51), \{t_4, t_2\})$
s_1: $((0.67, 0.72), \{t_4, t_3, t_2\}, 0.5, 0.8)$	m_2: $((0.19, 0.38), \{t_4, t_3, t_1\})$
s_2: $((0.88, 0.83), \{t_4, t_3, t_1\}, 0.5, 0.7)$	m_3: $((0.42, 0.40), \{t_3, t_2\})$
s_3: $((0.57, 0.89), \{t_5, t_4, t_1\}, 0.6, 0.75)$	m_4: $((0.49, 0.68), \{t_5, t_4, t_3\})$
s_4: $((0.25, 0.14), \{t_5, t_2, t_1\}, 0.5, 0.6)$	m_5: $((0.38, 0.80), \{t_5, t_4, t_2, t_1\})$
s_5: $((0.15, 0.32), \{t_3, t_2, t_1\}, 0.5, 0.6)$	m_6: $((0.11, 0.79), \{t_2\})$
s_6: $((0.10, 0.65), \{t_3, t_2, t_1\}, 0.2, 0.7)$	m_7: $((0.20, 0.70), \{t_4, t_1\})$
s_7: $((0.85, 0.04), \{t_5, t_4, t_1\}, 0.6, 0.7)$	
s_8: $((0.76, 0.12), \{t_5, t_4, t_2\}, 0.7, 0.7)$	
s_9: $((0.88, 0.24), \{t_4, t_2, t_1\}, 0.5, 0.8)$	
s_{10}: $((0.17, 0.77), \{t_5, t_3\}, 0.5, 0.8)$	

Keywords

Keywords	Discount	Adidas	NIKE	T-shirt	Shoes
ID	t_5	t_4	t_3	t_2	t_1
Weight	0.5	0.4	0.3	0.2	0.1

Figure 2: A running example.

When $s.\tau^{\mathrm{T}} \le 0$, a message m may be relevant to s no matter whether they share common keywords. To address this issue, for a subscription s, if $s.\tau \le 1 - s.\delta$, [2] introduced a virtual dummy keyword "$*$" with weight of 0 (i.e., $w(*) = 0$), and the *prefix* of s includes its keywords and "$*$".

Regarding the spatial filter, based on the *first match keyword* (denoted by t_i) between $\mathcal{SIG}(s)$ and m (i.e., m does not contain keywords before t_i in $\mathcal{SIG}(s)$), [2] estimated an upper textual similarity bound of m to s as follows:

$$\mathbf{UTB}(s \mid t_i) = \frac{\sum_{j=i}^{|s.text|} w(t_j)}{w(s.text)} \ge \mathbf{TSIM}(s, m). \quad (6)$$

Accordingly, [2] estimated a lower spatial similarity bound between s and m as follows:

$$\mathbf{LSB}(s \mid t_i) = \frac{s.\tau - s.\delta \cdot \mathbf{UTB}(s \mid t_i)}{1 - s.\delta} \le \mathbf{SSIM}(s, m). \quad (7)$$

For any message, if its spatial similarity to s is smaller than the lower spatial similarity bound $\mathbf{LSB}(s \mid t_i)$, the subscription s can be safely pruned.

Since given a subscription s and a message m, we do not know which keyword is their *first match keyword* (if they have), and the *first match keywords* for different messages to the subscription are different, for each keyword t in $\mathcal{SIG}(s)$, and [2] computed the lower spatial similarity bound $\mathbf{LSB}(s \mid t)$. This *prefix* of each subscription s with lower spatial similarity bound is called *spatial-oriented prefix*. If subscription s is relevant to message m, there must exist a keyword t in $\mathcal{SIG}(s) \cap m$ such that $\mathbf{SSIM}(s, m) \ge \mathbf{LSB}(s \mid t)$.

Based on the *spatial-oriented prefix*, [2] devised a filter-verification framework. In particular, an inverted index is built on the *spatial-oriented prefixes* first. Then, in the filter phase, for each message keyword t, the framework retrieves the inverted list $\mathcal{L}(t)$ of t and for each subscription s in $\mathcal{L}(t)$, if $\mathbf{SSIM}(s, m) \ge \mathbf{LSB}(s \mid t)$, s is put into the candidate set. In the verification phase, based on (1), the framework verifies whether each candidate s is an answer, and if yes, the message m is delivered to s.

3. Authenticated Location-Aware Publish/Subscribe Framework

In this section, we present our proposed authenticated location-aware publish/subscribe framework. In the publish/subscribe scenario, the messages delivered to the subscribers need to be verified as correct or not (i.e., *soundness* and *completeness*). However, compared with the subscriptions data, the messages data set is infinite, which can be regarded as the stream data. In such a situation, we (actually the data owner in the practical framework) cannot construct an authenticated data structure ($\mathcal{ADS}$) over the infinite messages data and, based on such a structure, construct the $\mathcal{VO}$ for subscribers' authentication. Therefore, intuitively, we need to sign every coming message and when the signed message is delivered to its corresponding subscribers, they can authenticate this message. However, when many messages need to be delivered to only one subscriber (the subscriber registers many interests in the framework), since every message has a signature, the communication cost between the service provider and this subscriber is high. Moreover, since the decryption of the signature is not a cheap operation, the authentication cost at the subscriber is also high. To tackle this problem, we present an authenticated location-aware publish/subscribe framework, which not only can deliver the messages more efficiently than the framework in the existing work [2], but also can make the subscribers' authentication available with low communication and authentication cost.

The main idea of our framework is to assume that the messages are allowed a maximum delay Δt to be delivered

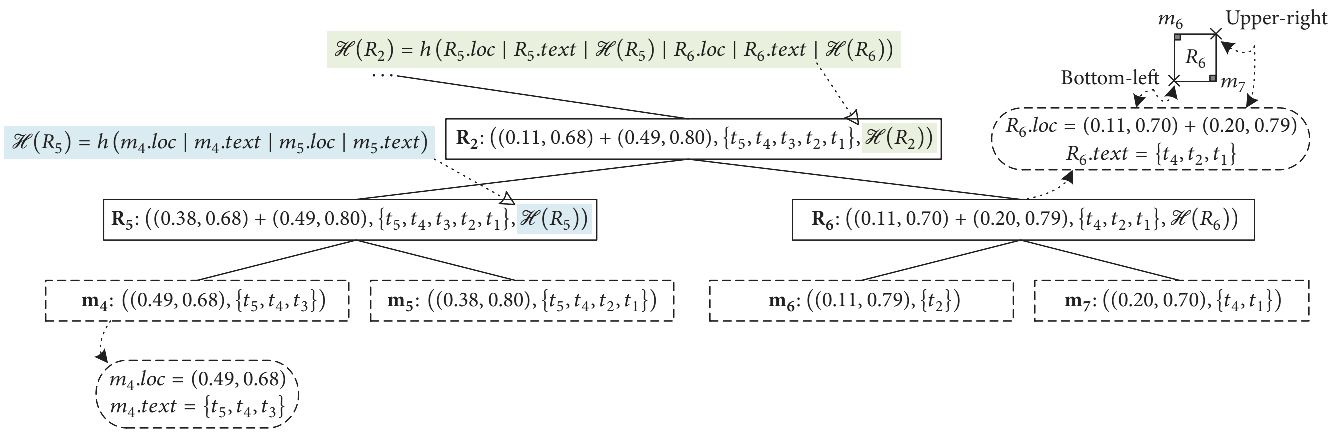

$\mathscr{H}(R_2) = h(R_5.loc \mid R_5.text \mid \mathscr{H}(R_5) \mid R_6.loc \mid R_6.text \mid \mathscr{H}(R_6))$
$\mathscr{H}(R_5) = h(m_4.loc \mid m_4.text \mid m_5.loc \mid m_5.text)$
$\mathbf{R_2}$: $((0.11, 0.68) + (0.49, 0.80), \{t_5, t_4, t_3, t_2, t_1\}, \mathscr{H}(R_2))$
$\mathbf{R_5}$: $((0.38, 0.68) + (0.49, 0.80), \{t_5, t_4, t_3, t_2, t_1\}, \mathscr{H}(R_5))$
$\mathbf{R_6}$: $((0.11, 0.70) + (0.20, 0.79), \{t_4, t_2, t_1\}, \mathscr{H}(R_6))$
$\mathbf{m_4}$: $((0.49, 0.68), \{t_5, t_4, t_3\})$
$\mathbf{m_5}$: $((0.38, 0.80), \{t_5, t_4, t_2, t_1\})$
$\mathbf{m_6}$: $((0.11, 0.79), \{t_2\})$
$\mathbf{m_7}$: $((0.20, 0.70), \{t_4, t_1\})$
$m_4.loc = (0.49, 0.68)$
$m_4.text = \{t_5, t_4, t_3\}$
$R_6.loc = (0.11, 0.70) + (0.20, 0.79)$
$R_6.text = \{t_4, t_2, t_1\}$
m_6
Upper-right
R_6
Bottom-left
m_7

FIGURE 3: Illustration of TMR-tree (partial).

to their corresponding subscribers. Under this circumstance, a batch of messages, rather than only one message, can be computed at a time. We organize these messages in a Merkle hash tree ($\mathscr{MHT}$) like structure (i.e., the $\mathscr{ADS}$). When more than one message is delivered to a subscriber, only one signature is returned, thereby reducing the communication and authentication cost. Moreover, recall that, in [2], an inverted index of *spatial-oriented prefixes* of all the subscriptions is constructed and when a message comes, the framework retrieves the inverted index to compute which subscription is relevant to this message. The message needs to be computed with every subscription in $\mathscr{L}(t)$ of every message keyword t. To reduce the computational cost and improve the efficiency of messages delivery, we present an inverted index pruning technique. By using the constraint of the spatial proximity between these messages (the messages are also organized in an R-tree like structure), we can prune some subscriptions which cannot become the delivery destinations from the inverted index and thus they need not be involved in the further computation.

3.1. Text-Aware Merkle R-Tree (TMR-Tree). We first introduce the method of constructing the $\mathscr{ADS}$, called Text-aware Merkle R-tree (TMR-tree), at the data owner side. Consider a predefined maximum permissible delivery delay Δt. The data owner builds one TMR-tree on all the messages within every time interval $\Delta t'$ ($\Delta t' \leq \Delta t$). Specifically, the TMR-tree has four main features:

(i) The messages in $\Delta t'$ are spatially organized in an R-tree.

(ii) Each node has a pseudo-text which includes the union of the keywords in its children's texts. A node u with children v and w has pseudo-text $u.text = v.text \cup w.text$.

(iii) Similar to the $\mathscr{MHT}$, the TMR-tree stores one hash value in each node. Assume the default fanout of the TMR-tree is 2. A leaf node u with children (messages) v and w stores hash value $\mathscr{H}(u) = h(v \mid w)$. An internal node u with children v and w stores hash value $\mathscr{H}(u) = h(v \mid \mathscr{H}(v) \mid w \mid \mathscr{H}(w))$, where $\mathscr{H}(v)$ and $\mathscr{H}(w)$ are the hash values of v and w. More specifically, the spatial and textual information of v (w) are both involved in the computation, that is, $\mathscr{H}(u) = h(v.loc \mid v.text \mid w.loc \mid w.text)$ if u is a leaf node in the TMR-tree and $\mathscr{H}(u) = h(v.loc \mid v.text \mid \mathscr{H}(v) \mid w.loc \mid w.text \mid \mathscr{H}(w))$ if u is an internal node in the TMR-tree.

(iv) The hash value of the root of the TMR-tree is signed by the data owner, producing signature $\mathscr{S}$.

Example 1. Figure 3 shows an example of the TMR-tree constructed over the messages in Figure 2. Here the fanout of the TMR-tree is set as 2. The leaf node R_5 has a pseudo-text $\{t_5, t_4, t_3, t_2, t_1\}$ which is the union of its children's texts, that is, $\{t_5, t_4, t_3\} \cup \{t_5, t_4, t_2, t_1\}$ ($m_4.text \cup m_5.text$). Since no matter the leaf or internal node is represented by a rectangle area (the Minimum Bounding Rectangle ($\mathscr{MBR}$) of messages or other $\mathscr{MBR}s$ in it), its location is defined by two points which can be the bottom-left and upper-right points. For example, R_6 is the $\mathscr{MBR}$ of m_6 and m_7 and its location includes the rectangle's bottom-left and upper-right points ((0.11, 0.70) and (0.20, 0.79)). The leaf node R_5 stores a hash value $\mathscr{H}(R_5)$ which summarizes the authentication information about R_5 and it is computed through the spatial and textual information of its children (messages); that is, $\mathscr{H}(R_5) = h(m_4.loc \mid m_4.text \mid m_5.loc \mid m_5.text)$. Similarly, the hash value stored in internal node R_2 is computed through the spatial and textual information of its children and their hash values; that is, $\mathscr{H}(R_2) = h(R_5.loc \mid R_5.text \mid \mathscr{H}(R_5) \mid R_6.loc \mid R_6.text \mid \mathscr{H}(R_6))$.

3.2. Filter-Verification Framework with Inverted Index Pruning. We use the idea of filter-verification framework proposed in [2] and, based on this, present an inverted index pruning technique to reduce the computational cost, thereby improving the efficiency of messages delivery.

Since a location and a pseudo-text are associated with a node in the TMR-tree, each node in the TMR-tree can be treated as a dummy message. The main idea of our inverted index pruning technique is based on the following proposition.

```
Input: TMR-tree: the text-aware Merkle R-tree on messages within $\Delta t'$; $\mathscr{L}$: the inverted index of spatial-oriented prefixes
Output: $\mathscr{A}m_i$: answers of each message $m_i$ in the TMR-tree
(1) TMR-tree.Root.$\mathscr{L}p = \mathscr{L}$;
(2) Initialize Stack $T$;
(3) $T$.Push(TMR-tree.Root);
(4) while $T$ is not empty do
(5)       $R = T$.Pop();
(6)       for each $t_i \in R.text$ do
(7)             for each $s_j \in R.\mathscr{L}p(t_i)$ do
(8)                   if SSIM$(s_j, R) \geq$ LSB$(s_j \mid t_i)$ then
(9)                         $R.\mathscr{L}r(t_i)$.Add$(s_j)$;
(10)      if $R.\mathscr{L}r \neq \emptyset$ and $R$ is not a message then
(11)            $T$.Push($R$.children);
(12) for each $m_i \in$ TMR-tree.leaf nodes do
(13)      for each $s_j \in m_i.\mathscr{L}r$ do
(14)            if Verify$(s_j, m_i)$ then
(15)                  $\mathscr{A}m_i$.Add$(s_j)$;
(16) return $\mathscr{A}m_1, \mathscr{A}m_2, \ldots, \mathscr{A}m_M$;
```

Algorithm 1: Filter-verification framework with inverted index pruning.

Proposition 2. *If some subscriptions in the inverted index of spatial-oriented prefixes ($\mathscr{L}$) are not relevant to a node R (a dummy message) in the TMR-tree, these subscriptions can be safely pruned from $\mathscr{L}$ since R's children (other dummy messages or true messages) cannot be relevant to these subscriptions with certainty.*

Proof. If a subscription s in $\mathscr{L}$ is not relevant to a node R, we have $\exists s \in \mathscr{L}$, $\mathbf{SSIM}(s, R) < \mathbf{LSB}(s \mid t)$. According to (3), we have

$$\max\left(0, 1 - \frac{\mathbf{DIST}(s.loc, R.loc)}{\mathbf{maxDIST}}\right) < \mathbf{LSB}(s \mid t). \quad (8)$$

$\forall R' \in R$, that is, R' is R's child, since $\mathbf{DIST}(s.loc, R'.loc) \geq \mathbf{DIST}(s.loc, R.loc)$, we have

$$\max\left(0, 1 - \frac{\mathbf{DIST}\left(s.loc, R'.loc\right)}{\mathbf{maxDIST}}\right) \leq \max\left(0, 1 - \frac{\mathbf{DIST}(s.loc, R.loc)}{\mathbf{maxDIST}}\right) < \mathbf{LSB}(s \mid t). \quad (9)$$

Therefore, s is not relevant to R' either. □

Algorithm 1 shows the pseudo-code of our framework with the inverted index pruning technique. It takes the TMR-tree and $\mathscr{L}$ as input. For each node R and each message m in leaf nodes in the TMR-tree, we use $R.\mathscr{L}r$ ($m.\mathscr{L}r$) to denote the inverted index where the subscriptions in $R.\mathscr{L}r$ ($m.\mathscr{L}r$) are likely to be relevant to R (m), which is pruned from $\mathscr{L}r$ of R's (m's) parent. We let $R.\mathscr{L}p$ ($m.\mathscr{L}p$) denote $\mathscr{L}r$ of R's (m's) parent. Thus, $\mathscr{L}r$ of each node (or message) is pruned from its $\mathscr{L}p$, which is its parent's $\mathscr{L}r$. In the beginning, we set the root's $\mathscr{L}p$ (i.e., TMR-tree.Root.$\mathscr{L}p$) as $\mathscr{L}$, which is the inverted index of *spatial-oriented prefixes* of all the subscriptions without any pruning (line (1)). Then, we initialize an empty stack T and push the root into it (lines (2)-(3)). In the filter phase, every element in T is computed until T is empty (lines (4)–(11)). First, we pop an element R from T (line (5)). Then, for each keyword t_i in $R.text$, we retrieve the inverted list $R.\mathscr{L}p(t_i)$ of t_i and for each subscription s_j in $R.\mathscr{L}p(t_i)$, if $\mathtt{SSIM}(s_j, R) \geq \mathbf{LSB}(s_j \mid t_i)$, s_j is added to $R.\mathscr{L}r(t_i)$ (lines (6)–(9)). Other subscriptions which do not satisfy the condition in line (8) are pruned from $R.\mathscr{L}p$. If $R.\mathscr{L}r \neq \emptyset$ and R is not a message, that is, there exist some subscriptions in $R.\mathscr{L}r$ which might be relevant to R's children, R's children are put into stack T (lines (10)-(11)). The subscriptions in each $m_i.\mathscr{L}r$ are the candidates which might be relevant to m_i. In the verification phase, for each message m_i, we verify whether each candidate s_j in $m_i.\mathscr{L}r$ is the answer of m_i and if yes, s_j is added to the answer set of m_i, that is, $\mathscr{A}m_i$ (lines (12)–(15)). After computing all the messages in the TMR-tree, all the answer sets ($\mathscr{A}m_1, \mathscr{A}m_2, \ldots, \mathscr{A}m_M$) are together returned (line (16)). Here we assume that there are M messages that come within $\Delta t'$. Each subscription in $\mathscr{A}m_i$ is the delivery destination of the message m_i.

Example 3. Figure 4 shows an example of procedures of our proposed inverted index pruning technique in filter-verification framework. In step ①, the root is popped from stack T first. Then, for each keyword in *Root.text* (t_1 to t_5 and $*$), we retrieve the corresponding inverted list in $Root.\mathscr{L}p$ to compute the spatial similarity between the root and each subscription in this inverted list. Take subscription s_9 as an example, we retrieve $Root.\mathscr{L}p(t_1)$ and compute the spatial similarity between the root and s_9: $\mathtt{SSIM}(s_9, Root)$, which equals 1. Since $\mathtt{SSIM}(s_9, Root) = 1$ is greater than $\mathbf{LSB}(s_9 \mid t_1) = 0.60$, s_9 is added to $Root.\mathscr{L}r(t_1)$. Notice that the values of spatial similarity between the root and s_0, s_1, s_9 in $Root.\mathscr{L}p(t_4)$ and s_3, s_7, s_8 in $Root.\mathscr{L}p(t_5)$ are smaller than the values of these subscriptions' $\mathbf{LSB}(s \mid t)$; that is, they do not satisfy the condition in line (8) in Algorithm 1.

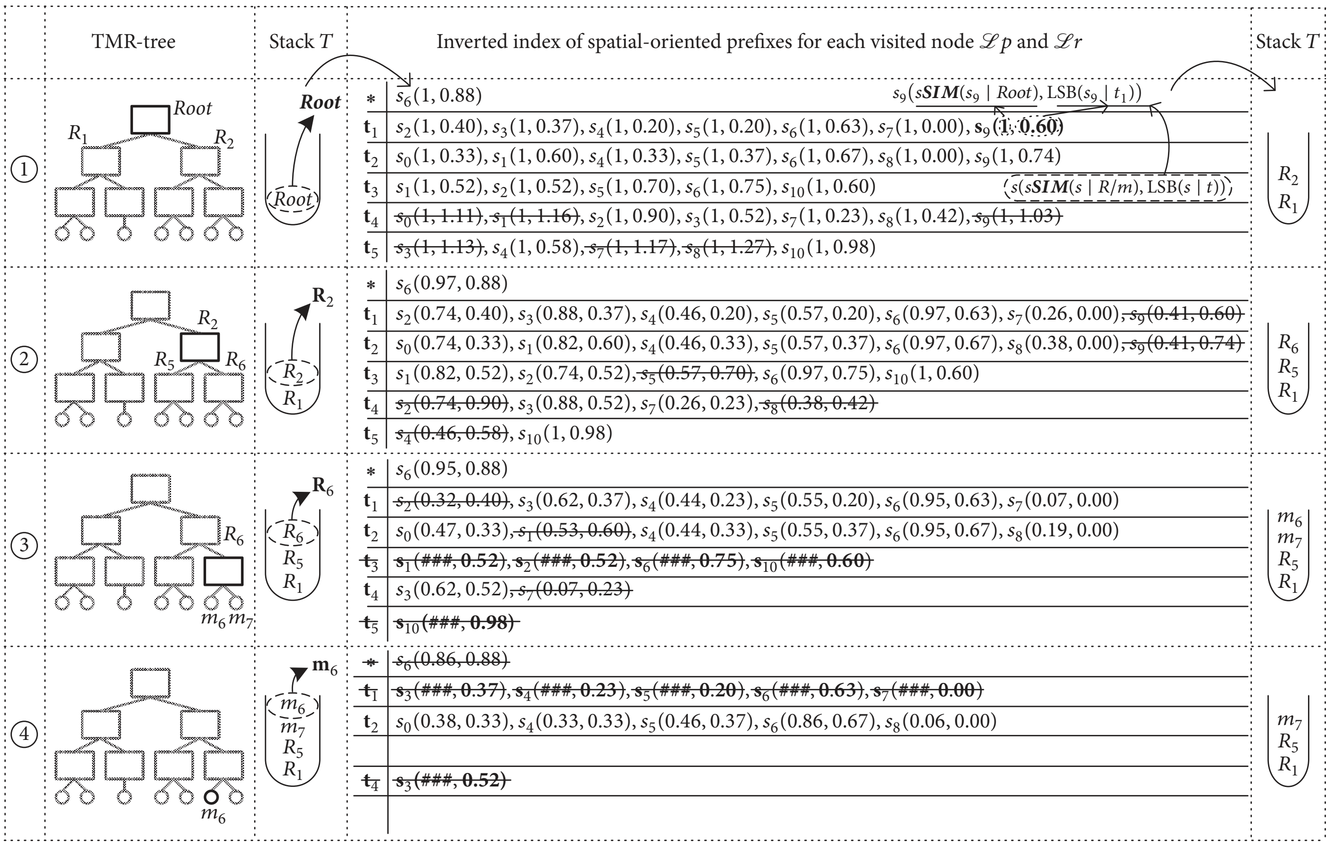

TMR-tree
Stack T
Inverted index of spatial-oriented prefixes for each visited node $\mathscr{L}p$ and $\mathscr{L}r$
Stack T
①
Root
R_1
R_2
Root
$*$ $s_6(1, 0.88)$
$s_9(s\boldsymbol{SIM}(s_9 \mid Root), \mathrm{LSB}(s_9 \mid t_1))$
t_1 $s_2(1, 0.40), s_3(1, 0.37), s_4(1, 0.20), s_5(1, 0.20), s_6(1, 0.63), s_7(1, 0.00), \mathbf{s_9(1, 0.60)}$
t_2 $s_0(1, 0.33), s_1(1, 0.60), s_4(1, 0.33), s_5(1, 0.37), s_6(1, 0.67), s_8(1, 0.00), s_9(1, 0.74)$
t_3 $s_1(1, 0.52), s_2(1, 0.52), s_5(1, 0.70), s_6(1, 0.75), s_{10}(1, 0.60)$
$s(s\boldsymbol{SIM}(s \mid R/m), \mathrm{LSB}(s \mid t))$
t_4 ~~$s_0(1, 1.11)$~~, ~~$s_1(1, 1.16)$~~, $s_2(1, 0.90), s_3(1, 0.52), s_7(1, 0.23), s_8(1, 0.42)$, ~~$s_9(1, 1.03)$~~
t_5 ~~$s_3(1, 1.13)$~~, $s_4(1, 0.58)$, ~~$s_7(1, 1.17)$~~, ~~$s_8(1, 1.27)$~~, $s_{10}(1, 0.98)$
R_2
R_1
②
R_2
R_5
R_6
R_2
R_1
$*$ $s_6(0.97, 0.88)$
t_1 $s_2(0.74, 0.40), s_3(0.88, 0.37), s_4(0.46, 0.20), s_5(0.57, 0.20), s_6(0.97, 0.63), s_7(0.26, 0.00)$, ~~$s_9(0.41, 0.60)$~~
t_2 $s_0(0.74, 0.33), s_1(0.82, 0.60), s_4(0.46, 0.33), s_5(0.57, 0.37), s_6(0.97, 0.67), s_8(0.38, 0.00)$, ~~$s_9(0.41, 0.74)$~~
t_3 $s_1(0.82, 0.52), s_2(0.74, 0.52)$, ~~$s_5(0.57, 0.70)$~~, $s_6(0.97, 0.75), s_{10}(1, 0.60)$
t_4 ~~$s_2(0.74, 0.90)$~~, $s_3(0.88, 0.52), s_7(0.26, 0.23)$, ~~$s_8(0.38, 0.42)$~~
t_5 ~~$s_4(0.46, 0.58)$~~, $s_{10}(1, 0.98)$
R_6
R_5
R_1
③
R_6
m_6 m_7
R_6
R_5
R_1
$*$ $s_6(0.95, 0.88)$
t_1 ~~$s_2(0.32, 0.40)$~~, $s_3(0.62, 0.37), s_4(0.44, 0.23), s_5(0.55, 0.20), s_6(0.95, 0.63), s_7(0.07, 0.00)$
t_2 $s_0(0.47, 0.33)$, ~~$s_1(0.53, 0.60)$~~, $s_4(0.44, 0.33), s_5(0.55, 0.37), s_6(0.95, 0.67), s_8(0.19, 0.00)$
~~t_3~~ ~~$s_1(\#\#\#, 0.52)$, $s_2(\#\#\#, 0.52)$, $s_6(\#\#\#, 0.75)$, $s_{10}(\#\#\#, 0.60)$~~
t_4 $s_3(0.62, 0.52)$, ~~$s_7(0.07, 0.23)$~~
~~t_5~~ ~~$s_{10}(\#\#\#, 0.98)$~~
m_6
m_7
R_5
R_1
④
m_6
m_6
m_7
R_5
R_1
~~$*$~~ ~~$s_6(0.86, 0.88)$~~
~~t_1~~ ~~$s_3(\#\#\#, 0.37)$, $s_4(\#\#\#, 0.23)$, $s_5(\#\#\#, 0.20)$, $s_6(\#\#\#, 0.63)$, $s_7(\#\#\#, 0.00)$~~
t_2 $s_0(0.38, 0.33), s_4(0.33, 0.33), s_5(0.46, 0.37), s_6(0.86, 0.67), s_8(0.06, 0.00)$
~~t_4~~ ~~$s_3(\#\#\#, 0.52)$~~
m_7
R_5
R_1

FIGURE 4: An Example of procedures of filter-verification framework with inverted index pruning (partial).

Therefore, they are pruned from $Root.\mathscr{L}p$ and are not added to $Root.\mathscr{L}r$. Then, since $Root.\mathscr{L}r \neq \emptyset$, its children R_1 and R_2 are pushed into stack T. Similarly, in step ②, s_9 in $R_2.\mathscr{L}p(t_1)$, s_9 in $R_2.\mathscr{L}p(t_2)$, s_5 in $R_2.\mathscr{L}p(t_3)$, s_2, s_8 in $R_2.\mathscr{L}p(t_4)$, and s_4 in $R_2.\mathscr{L}p(t_5)$ are pruned from $R_2.\mathscr{L}p$. Note that, in step ③, since $R_6.text$ does not contain the keywords t_3 and t_5, we need not compute the spatial similarity between R_6 and each subscription in inverted lists $R_6.\mathscr{L}p(t_3)$ and $R_6.\mathscr{L}p(t_5)$ (indicated by "###" in the example), and these subscriptions are also pruned from $R_6.\mathscr{L}p$. At last, in step ④, the candidates which might be relevant to m_6 are s_0, s_4, s_5, s_6, and s_8. Compared with all the subscriptions in the unpruned inverted index $\mathscr{L}$, the computational cost is reduced dramatically.

Time Complexity. For the convenience of comparison between the state-of-the-art method for location-aware publish/subscribe and our proposed filter-verification framework with inverted index pruning technique, we first give the time complexity of filter-verification framework proposed in [2] as the following proposition.

Proposition 4. *The time complexity of delivering one message m to its relevant subscriber s in filter-verification framework proposed in [2] is $\mathcal{O}(\sum_{t \in m.text} |\mathscr{L}(t)| + |s.text|)$, where $\mathscr{L}$ is the inverted index built on the spatial-oriented prefixes.*

Proof. In the filter phase, for each keyword t in $m.text$ (including the dummy keyword "$*$"), the framework retrieves the inverted list $\mathscr{L}(t)$ of t and for each subscription s in $\mathscr{L}(t)$, if $\mathbf{SSIM}(s, m) \geq \mathbf{LSB}(s \mid t)$, s is a candidate to the message m and we add it into the candidate set. Therefore, the time complexity of filtering is $\mathcal{O}(\sum_{t \in m.text} |\mathscr{L}(t)|)$.

In the verification phase, based on the spatio-textual similarity function (see (1)), s and m are relevant, if and only if

$$w(s.text \cap m.text) = \sum_{t \in s.text \cap m.text} w(t) \geq \frac{s.\tau - (1 - s.\delta) \cdot \mathbf{SSIM}(s, m)}{s.\delta} \cdot w(s.text). \quad (10)$$

$\mathbf{SSIM}(s, m)$ can be easily computed in $\mathcal{O}(1)$ time and $w(s.text)$ can be materialized; thus it is easy to compute $((s.\tau - (1 - s.\delta) \cdot \mathbf{SSIM}(s, m))/(s.\delta)) \cdot w(s.text)$. To compute $w(s.text \cap m.text)$, we check whether each keyword t in $s.text$ appears in $m.text$. If yes, we add the corresponding weight $w(t)$ into $w(s.text \cap m.text)$. To facilitate the checking, we build a hash map for keywords in $m.text$. (We only need to build the hash map for the message once.) Thus the time complexity of verifying a subscription s is $\mathcal{O}(|s.text|)$. Therefore, the

```
    **Input**: TMR-tree: a text-aware Merkle R-tree on messages within $\Delta t'$; $\mathscr{A}m_i$: answers of each message $m_i$ in the TMR-tree
    **Output**: $\mathscr{V}\mathscr{O}s_i$: the $\mathscr{V}\mathscr{O}$ for each subscription $s_i$
(1) for each $s_i \in \mathscr{A}m_1 \cup \mathscr{A}m_2 \cup \cdots \cup \mathscr{A}m_M$ do
(2)     $\mathscr{V}\mathscr{O}s_i$.Init(TMR-tree.Root);
(3) Initialize Queue Q;
(4) Q.Put(TMR-tree.Root);
(5) while $Q$ is not empty do
(6)     $R$ = Q.Pick();
(7)     for each $m_i \in$ TMR-tree.leaf nodes do
(8)         if Dist($m_i$, $R$) $\leq 0$ then
(9)             for each $s_j \in \mathscr{A}m_i$ do
(10)                $\mathscr{V}\mathscr{O}s_j$.Replace($R$, "[" + $R$.children + "]");
(11)    if $R$.children are not in $Q$ and they are not messages then
(12)        Q.Put($R$.children);
(13) return $\mathscr{V}\mathscr{O}s_1, \mathscr{V}\mathscr{O}s_2, \ldots, \mathscr{V}\mathscr{O}s_N$;
```

Algorithm 2: $\mathscr{V}\mathscr{O}$ construction.

time complexity of delivering one message m to its relevant subscriber s in filter-verification framework proposed in [2] is $\mathcal{O}(\sum_{t \in m.text} |\mathscr{L}(t)| + |s.text|)$. □

The time complexity of our proposed filter-verification framework with inverted index pruning technique is given by the following proposition.

Proposition 5. *The time complexity of delivering M messages that come within $\Delta t'$ to their relevant subscribers in our proposed filter-verification framework with inverted index pruning technique is $\mathcal{O}(\sum_{i=1}^{(1-M)/(1-f)} \sum_{t \in R_i.text} |R_i.\mathscr{L}p(t)| + \sum_{i=1}^{M} |s_i.text|)$, where f is the fanout of the TMR-tree, R_i is a node in the TMR-tree (also can be treated as a dummy message), and $\mathscr{L}p$ is the inverted index associated with R_i's parent in which the subscriptions are likely to be relevant to R_i's parent.*

Proof. A TMR-tree is constructed over the M messages that come within $\Delta t'$. If the fanout of the TMR-tree is f, in the worst case, the height of the TMR-tree (excluding the layer of messages) is $h = \log_f^M$. Thus, the number of internal and leaf nodes in the TMR-tree (assuming the root has depth 1) is

$$f^0 + \cdots + f^{h-1} = \frac{1-f^h}{1-f} = \frac{1-f^{\log_f^M}}{1-f} = \frac{1-M}{1-f}. \quad (11)$$

In the filter phase, when visiting a node R (or a message m) in the TMR-tree, we retrieve its $\mathscr{L}p$, prune $R.\mathscr{L}p$ (or $m.\mathscr{L}p$), and generate $R.\mathscr{L}r$ (or $m.\mathscr{L}r$). Therefore, the time complexity of filtering with inverted index pruning is $\mathcal{O}(\sum_{i=1}^{(1-M)/(1-f)} \sum_{t \in R_i.text} |R_i.\mathscr{L}p(t)|)$.

In the verification phase, suppose each message m within $\Delta t'$ is delivered to only one subscriber. The time complexity of verifying whether M subscriptions are the answers of messages is $\mathcal{O}(\sum_{i=1}^{M} |s_i.text|)$. Therefore, the time complexity of delivering M messages that come within $\Delta t'$ to their relevant subscribers in our proposed filter-verification framework with inverted index pruning technique is $\mathcal{O}(\sum_{i=1}^{(1-M)/(1-f)} \sum_{t \in R_i.text} |R_i.\mathscr{L}p(t)| + \sum_{i=1}^{M} |s_i.text|)$. □

Compared with the filter-verification framework proposed in [2], our proposed filter-verification framework with inverted index pruning technique needs to visit more inverted indexes (the inverted indexes associated with internal and leaf nodes in the TMR-tree). However, since the subscriptions in each node's inverted index are constantly pruned from the root to the leaf nodes in the TMR-tree, the total times of inverted index traversal are reduced. Therefore, our proposed filter-verification framework with inverted index pruning technique can be considered efficient, which also can be demonstrated from our experimental study (Section 4).

3.3. $\mathscr{V}\mathscr{O}$ Construction and Authentication. After finding the subscribers who are the delivery destinations of messages, that is, $\mathscr{A}m_i$, the service provider still needs to construct a $\mathscr{V}\mathscr{O}$ for each subscriber for their authentication. Algorithm 2 shows the pseudo-code of constructing the $\mathscr{V}\mathscr{O}$. It takes the TMR-tree and answers of each message m_i ($\mathscr{A}m_i$) as input. First, we initialize a $\mathscr{V}\mathscr{O}$ for each subscription in $\mathscr{A}m_1 \cup \mathscr{A}m_2 \cup \cdots \cup \mathscr{A}m_M$ ($\mathscr{V}\mathscr{O}s_i$) with the root of the TMR-tree (lines (1)-(2)). Then, we initialize an empty queue Q and put the root into it (lines (3)-(4)). Every element in Q is computed until Q is empty (lines (5)–(12)). When the distance between a message m_i and the picked element R (from Q) is smaller than 0, that is, m_i is in the subtree rooted at R, for each subscription s_i in $\mathscr{A}m_i$, we replace R in $\mathscr{V}\mathscr{O}s_i$ with three parts: (1) the token "["; (2) R's children; and (3) the token "]" (lines (6)–(10)). Note that we use a pair of tokens "[" and "]" to indicate the scope of the entries in R. Then, if R's children are not in Q and they are not messages, they are put into Q (lines (11)-(12)). At last, the constructed $\mathscr{V}\mathscr{O}s_i$ is delivered to each subscriber s_i with the corresponding messages (line (13)). Here we assume

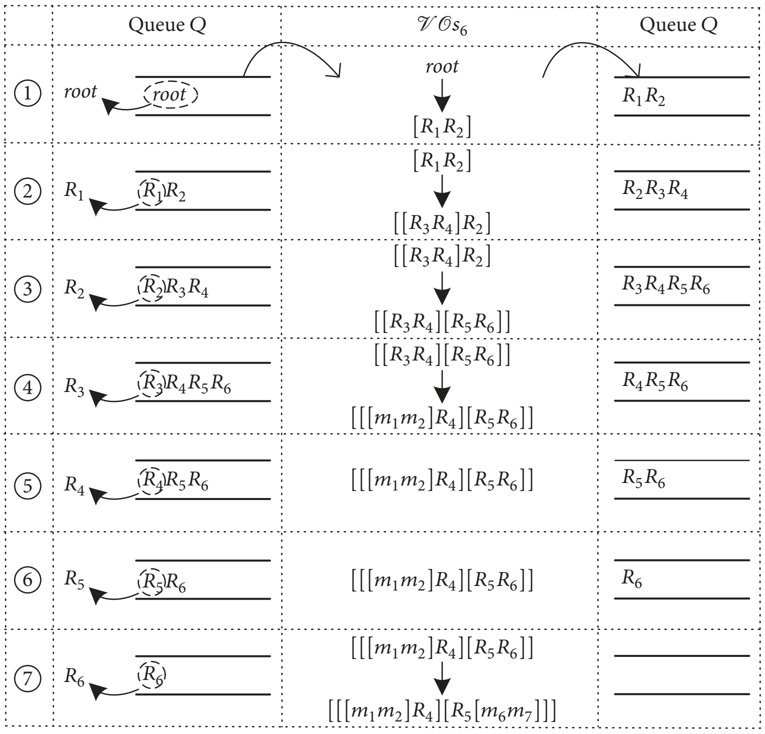

Queue Q
$\mathscr{VO}s_6$
Queue Q
① root root root $[R_1R_2]$ R_1R_2
② R_1 R_1R_2 $[R_1R_2]$ $[[R_3R_4]R_2]$ $R_2R_3R_4$
③ R_2 $R_2R_3R_4$ $[[R_3R_4]R_2]$ $[[R_3R_4][R_5R_6]]$ $R_3R_4R_5R_6$
④ R_3 $R_3R_4R_5R_6$ $[[R_3R_4][R_5R_6]]$ $[[[m_1m_2]R_4][R_5R_6]]$ $R_4R_5R_6$
⑤ R_4 $R_4R_5R_6$ $[[[m_1m_2]R_4][R_5R_6]]$ R_5R_6
⑥ R_5 R_5R_6 $[[[m_1m_2]R_4][R_5R_6]]$ R_6
⑦ R_6 R_6 $[[[m_1m_2]R_4][R_5R_6]]$ $[[[m_1m_2]R_4][R_5[m_6m_7]]]$

FIGURE 5: An example of procedures of $\mathscr{VO}$ construction ($\mathscr{VO}s_6$).

that there are N subscribers to whom the messages will be delivered.

Example 6. Following the example in Figure 2, after computing the delivery destinations of messages m_1 to m_7, we obtain their answer sets as follows: $\mathscr{A}m_1 : s_6$, $\mathscr{A}m_2 : s_5, s_6$, $\mathscr{A}m_3 : s_5$, $\mathscr{A}m_4 : s_2, s_3, s_7, s_{10}$, $\mathscr{A}m_5 : s_0, s_3, s_4, s_7, s_8$, $\mathscr{A}m_6 : s_6$, and $\mathscr{A}m_7 : s_6$. Here we take the construction of the $\mathscr{VO}$ of s_6 ($\mathscr{VO}s_6$) as an example (shown in Figure 5) and messages m_1, m_2, m_6, and m_7 will be delivered to s_6. In step ①, the root is picked from the queue Q. Obviously, **DIST**(m_1.*loc*, *root.loc*), **DIST**(m_2.*loc*, *root.loc*), **DIST**(m_6.*loc*, *root.loc*), and **DIST**(m_7.*loc*, *root.loc*) are all smaller than 0. Therefore, we replace "root" in $\mathscr{VO}s_6$ with "$[R_1R_2]$". Since R_1 and R_2 are not in Q and they are not messages, they are put into Q. The steps ②, ③, and ④ are similar to ①. In step ⑤, since m_1, m_2, m_6, and m_7 are not in the subtree root at R_4, that is, **DIST**(m_1.*loc*, R_4.*loc*), **DIST**(m_2.*loc*, R_4.*loc*), **DIST**(m_6.*loc*, R_4.*loc*), and **DIST**(m_7.*loc*, R_4.*loc*) are all greater than 0, "R_4" in $\mathscr{VO}s_6$ is not replaced and thus $\mathscr{VO}s_6$ remains unchanged. Due to the similar reason, in step ⑥, $\mathscr{VO}s_6$ still remains unchanged. At last, in step ⑦, after computing the last element R_6 in Q, $\mathscr{VO}s_6$ is "$[[[m_1m_2]R_4][R_5[m_6m_7]]]$".

To authenticate the *soundness* of delivered messages, each subscriber s_i needs to scan their $\mathscr{VO}s_i$ to recompute the hash value of the root of the TMR-tree and compare it against the root signature using the data owner's public key distributed by the $\mathscr{KDC}$. Since each $\mathscr{VO}s_i$ includes the entries which have been visited during messages delivery, the subscriber can simulate the procedure of the TMR-tree traversal and recursively reconstruct each $\mathscr{MBR}$ and compute its hash value in a bottom-up manner. Specifically, each $\mathscr{MBR}$ and its hash value can be computed from the entries in its child node which are indicated by "[" and "]".

To authenticate the *completeness* of delivered messages, the subscriber s_i needs to check that each message in results is indeed present in $\mathscr{VO}s_i$ and whether they satisfy the parameters δ and τ. What is more, the subscriber still needs to check that the other entries returned in the $\mathscr{VO}s_i$ do not satisfy δ and τ.

Example 7. Still taking s_6 as an example, the subscriber can recursively reconstruct R_3 from m_1 and m_2, R_1 from R_3 and R_4, R_6 from m_6 and m_7, R_2 from R_5 and R_6, and at last the root from R_1 and R_2 and compute its hash value to compare it against the root signature to authenticate the *soundness* of delivered messages m_1, m_2, m_6, and m_7. As for authenticating the *completeness* of m_1, m_2, m_6, and m_7, the subscriber needs to recompute whether they satisfy $s_6.\delta = 0.2$ and $s_6.\tau = 0.7$, while R_4 and R_5 do not.

From the example we can see that when more than one message is delivered to a subscriber, only one signature is returned, thus reducing the communication and authentication cost.

Space and Time Complexity. We first give a baseline method for the problem of authenticating messages in outsourced location-aware publish/subscribe services. Then, we give the space complexity of its $\mathcal{VO}$ and compare it with our proposed authenticated location-aware publish/subscribe framework. We also compare the authentication's time complexity of baseline method and our framework.

> *Baseline*: the data owner signs every message within $\Delta t'$ and when the signed messages are delivered to their corresponding subscribers, each $\mathcal{VO}s_i$ consists of the messages (to s_i) and their signatures. Then, the subscriber s_i can verify the *soundness* by computing the hash value of each message in $\mathcal{VO}s_i$ and comparing it against the message's signature. Recomputing the spatio-textual similarity between each message in $\mathcal{VO}s_i$ and the subscription s_i enables the subscriber to verify the *completeness*.

The space complexity of the $\mathcal{VO}$ of baseline method is given by following proposition.

Proposition 8. *If there are p messages which are delivered to a subscriber s at one time, the $\mathcal{VO}$ size for s, that is, the space complexity of $\mathcal{VO}s$, is $\mathcal{O}(\sum_{i=1}^{p}|m_i| + p|\mathcal{S}|)$, where $|\mathcal{S}|$ is the size of the signature and each $|m_i|$ includes the size of its spatial and textual information.*

Compared with the baseline method, the space complexity of $\mathcal{VO}$ of our proposed authenticated location-aware publish/subscribe framework is given by the following proposition.

Proposition 9. *If there are p messages which are delivered to a subscriber s at one time, the space complexity of $\mathcal{VO}s$ is $\mathcal{O}(\sum_{i=1}^{p}|m_i| + \sum_{i=1}^{q}|R_i| + |\mathcal{S}|)$, where $|\mathcal{S}|$ is the size of the signature. R_i is a dummy message and we assume there are q dummy messages that are included in $\mathcal{VO}s$.*

From the above propositions we can see that, in our proposed filter-verification framework with inverted index pruning technique, if more than one message is delivered to a subscriber, only one signature is returned. Although our framework has q dummy messages in its $\mathcal{VO}$, its $\mathcal{VO}$ size is still smaller than that of the baseline method when p is large since the signatures are space consuming.

Since the authentication time is co-related to the size of $\mathcal{VO}$, the time complexity of authentication of our proposed filter-verification framework with inverted index pruning technique is also smaller than that of the baseline method.

4. Experimental Study

In this section, we proceed to conduct extensive experiments to evaluate the performance of our proposed authenticated location-aware publish/subscribe framework.

4.1. Experiment Setup

4.1.1. Datasets. Similar to [2], we use a real-world dataset POI which contains 10 million points of interests in USA. We randomly select 1–5 keywords from each POI to generate subscriptions. Thus the average keyword number in each subscription is 3. The maximum permissible response delay Δt and the messages delivery interval $\Delta t'$ ($\Delta t' \leq \Delta t$) are both set as 5 mins. During this interval, we randomly select 2000 POIs as messages. To generate long messages, we combine 10 POIs as a single message. The average keyword number in each message is 41.

4.1.2. Parameters. The performance of our proposed framework is evaluated by varying the preference $s.\delta$ (0.1, 0.3, 0.5, 0.7, and 0.9) and threshold $s.\tau$ (0.5, 0.6, 0.7, 0.8, and 0.9). We set $s.\delta$ as 0.5 and $s.\tau$ as 0.7 in the default setting. When we vary a parameter, the other parameter will be in the default setting. We use inverted document frequency ($\mathcal{IDF}$) to generate keywords weights.

4.1.3. System Configuration. All the experiments are run on a server with Intel(R) Xeon(R) CPU E5-2609 v2 @2.5 GHz (Quad Core) and 64 GB RAM, running Linux Ubuntu. We use in-memory setting and the programs are implemented in C++.

4.1.4. Performance Metrics. The metrics for performance evaluation include

(i) PAS and PC: percentage of accessed subscriptions and candidates, which indicate the ratios of accessed subscriptions in the inverted index of *spatial-oriented prefixes* and candidates to the number of total subscriptions

(ii) FS: time of finding the relevant subscriptions for each message within $\Delta t'$

(iii) CVO: time of constructing the $\mathcal{VO}$

(iv) VOS: $\mathcal{VO}$ size, which affects the communication cost between the service provider and subscribers

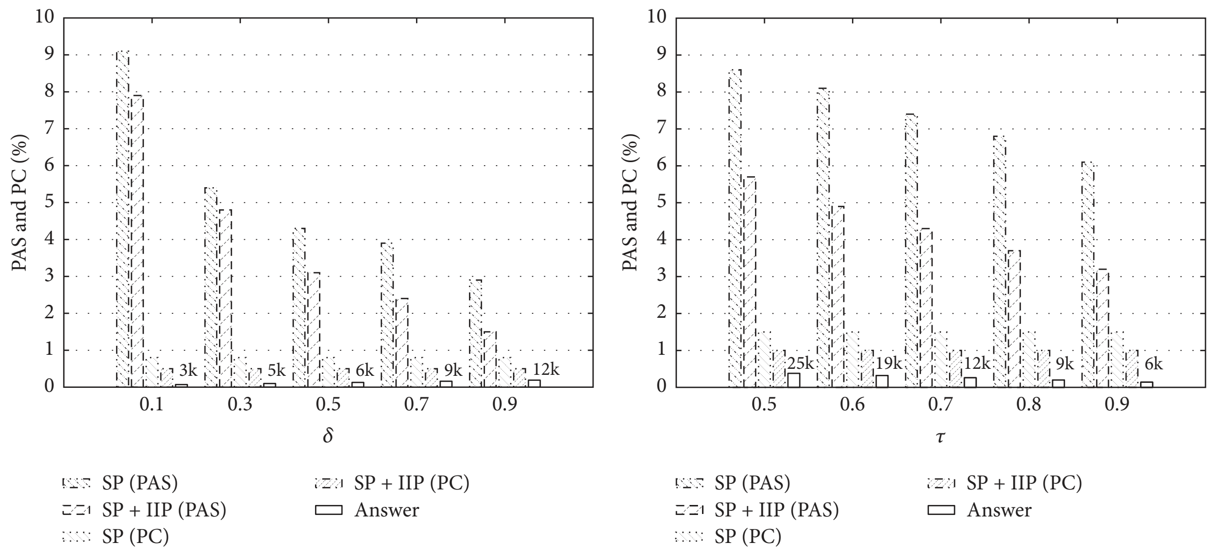
(v) AM: time of authenticating the messages at the subscribers side

Note that, in our framework, we process a batch of messages at one time; thus each time we first get a total value of each metric. Then, for the metrics PAS, PC, and FS, we report the average value corresponding to each message and, for the metrics CVO, VOS, and AM, we report the average value corresponding to each subscriber.

4.1.5. Algorithms. For metrics (i), (ii), and (iii), algorithms to be evaluated in our experiments include (1) SP (the method of finding the relevant subscriptions for each message using the *spatial-oriented prefixes*, which is proposed in [2]); (2) SP + IIP (our filter-verification framework with inverted index pruning technique); (3) VOC (our method of constructing the $\mathcal{VO}$).

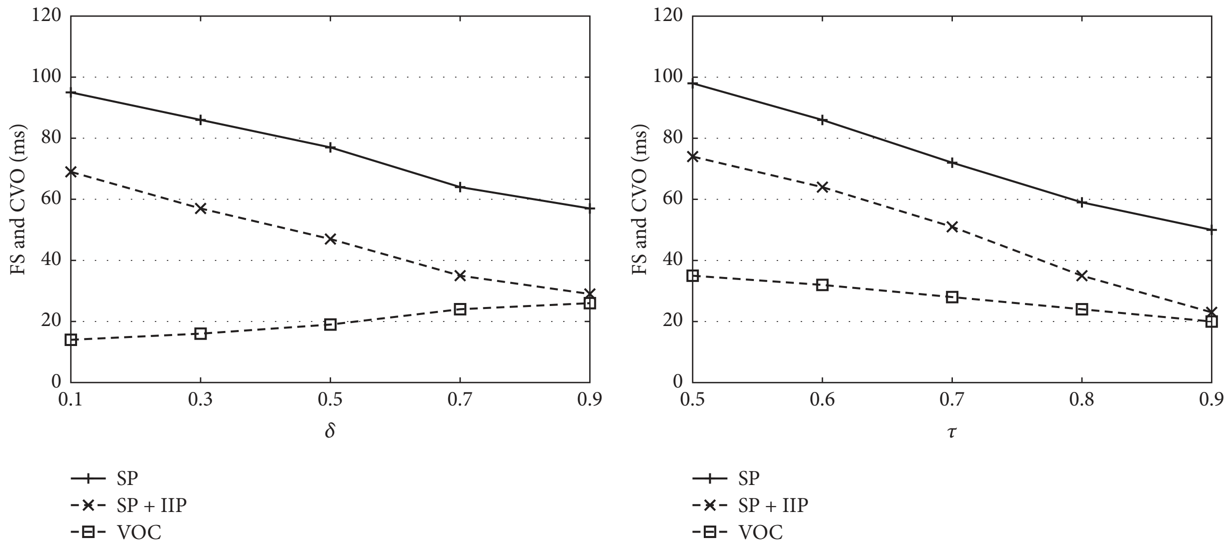
For metrics (iv) and (v), algorithms to be evaluated include (1) ALPF (our authenticated location-aware publish/subscribe framework) and (2) BL (the baseline method).

Note that, to the best of our knowledge, this is the first attempt to define and solve the problem of authenticating messages in outsourced location-aware publish/subscribe services. Therefore, no existing algorithm is included in our experiments as comparative analysis.



(a) Varying s_δ (b) Varying s_τ

Figure 6: Evaluation of PAS and PC.



(a) Varying s_δ (b) Varying s_τ

Figure 7: Evaluation of FS and CVO.

4.2. Performance Study

4.2.1. Cost at the Service Provider. The cost at the service provider is evaluated from two aspects. First, in Figure 6, we evaluate the ratios of accessed subscriptions and candidates (as a function of $s.\delta$ and $s.\tau$) to the number of total subscriptions (PAS and PC), where the accessed subscriptions refer to subscriptions that are accessed in the inverted index and candidates refer to subscriptions that are verified using the `Verify` function in Algorithm 1. Second, as shown in Figure 7, we evaluate the running time (as a function of $s.\delta$ and $s.\tau$), which includes the time of finding the relevant subscriptions for each message (FS) and constructing the $\mathscr{VO}$ (CVO).

According to Figures 6 and 7, we make the following observations. First, SP + IIP outperforms SP; that is, the PAS and PC of SP + IIP are both smaller than those of SP (shown in Figure 6). Besides, FS of SP + IIP is smaller than that of SP (shown in Figure 7). The reason lies in that SP + IIP uses the inverted index pruning technique to prune the subscriptions from the inverted index of *spatial-oriented prefixes*. These pruned subscriptions are not relevant to the messages and thus they need not be involved in the computation. Second, with the increase of $s.\tau$, the performance of SP and SP

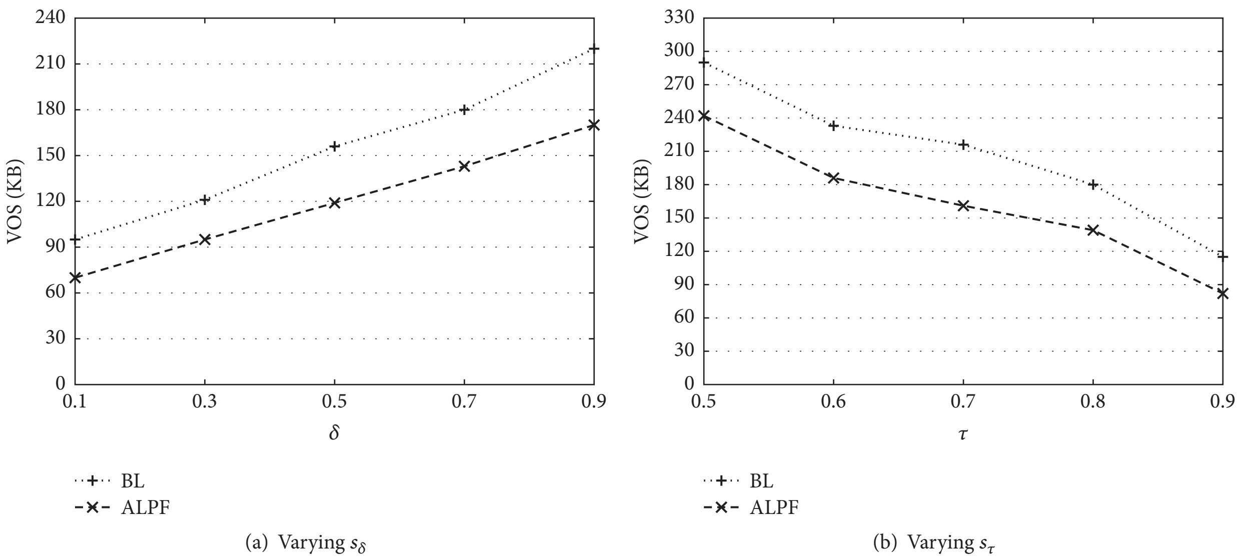


Figure 8: Evaluation of VOS.

+ IIP increases, because for larger $s.\tau$ there are smaller number of subscriptions required to be visited and verified, and we have greater opportunity to prune more irrelevant subscriptions. Third, with the decrease of $s.\delta$, SP and SP + IIP take much longer time, because for smaller $s.\delta$, the spatial similarity is more important and they cannot estimate accurate prefix bounds. Fourth, as $s.\delta$ ($s.\tau$) increases, CVO increases (decreases) slightly since we use the answers of each message to construct the $\mathcal{VO}$ and CVO depending on the number of answers. With the increase of $s.\delta$ ($s.\tau$), we get more (less) answers and thus CVO increases (decreases). Fifth, although in our framework it costs extra time to construct $\mathcal{VO}$ for subscribers' authentication, the total running time (FS + CVO) is still better than SP. For example, in Figure 7, when $s.\delta = 0.3$, SP + IIP costs around 60 ms and VOC costs about 20 ms, thus the total running time is about 80 ms, which is still less than the cost of SP, 90 ms.

4.2.2. Cost between the Service Provider and Subscribers. We evaluate the metric VOS, that is, $\mathcal{VO}$ size, which affects the communication overhead between the service provider and subscribers. Figure 8 shows VOS under the experimental settings by varying $s.\delta$ and $s.\tau$.

From Figure 8, we make the following observations. First, ALPF outperforms BL since we process a batch of messages rather than only one message at a time and when many messages are delivered to a subscriber s_i, the $\mathcal{VO}s_i$ consists of only one signature, which is computed using the root hash value of the TMR-tree. However, in BL, the $\mathcal{VO}s_i$ would include the signatures of every message. Second, with the increase of $s.\delta$ ($s.\tau$), VOS increases (decreases) in a near linear manner. The reason lies in that VOS depends on the number of messages delivered to each subscriber. When $s.\delta$ ($s.\tau$) increases, the number of answers of each message increases (decreases) and, conversely, the number of messages delivered to each subscriber increases (decreases). Third, the biggest value of VOS is about 240 KB when $s.\tau = 0.5$. This value is acceptable especially when more than one message needs to be verified by a subscriber.

4.2.3. Cost at the Subscribers. The last metric AM, that is, the time of authenticating the messages at the subscribers side, is evaluated. AM is crucial since the subscribers may have limited computing resources. Figure 9 shows AM as a function of $s.\delta$ and $s.\tau$.

According to Figure 9, we first find that, in ALPF, it always costs subscribers less time to authenticate the messages delivered to them than that in BL. This is because in ALPF when the *soundness* is verified, subscribers just need to decrypt one signature and recompute the root hash value of the TMR-tree to compare against it. However, in BL, the number of decryption operations equals the number of messages delivered to the subscribers but decryption is not a cheap operation comparing with the hashing operation. Thus, ALPF outperforms BL. Second, we find that, with the increase of $s.\delta$ ($s.\tau$), AM increases (decreases) in a near linear manner since AM is always related to VOS and they have the same changing situation. Third, the worst case of authenticating the messages costs subscribers about 1.2 s, which is reasonable and would not have too many bad effects on the subscribers experience.

4.2.4. Security Analysis. In this paper, we study the problem of authenticating messages in outsourced location-aware publish/subscribe services. Therefore, our goal of security analysis is to prove that our proposed authenticated location-aware publish/subscribe framework can guarantee the verification of *soundness* and *completeness* of messages by their corresponding subscribers.

Proof of Soundness. Assume that a message m delivered to a subscriber s is bogus or modified. In this paper, we adopt

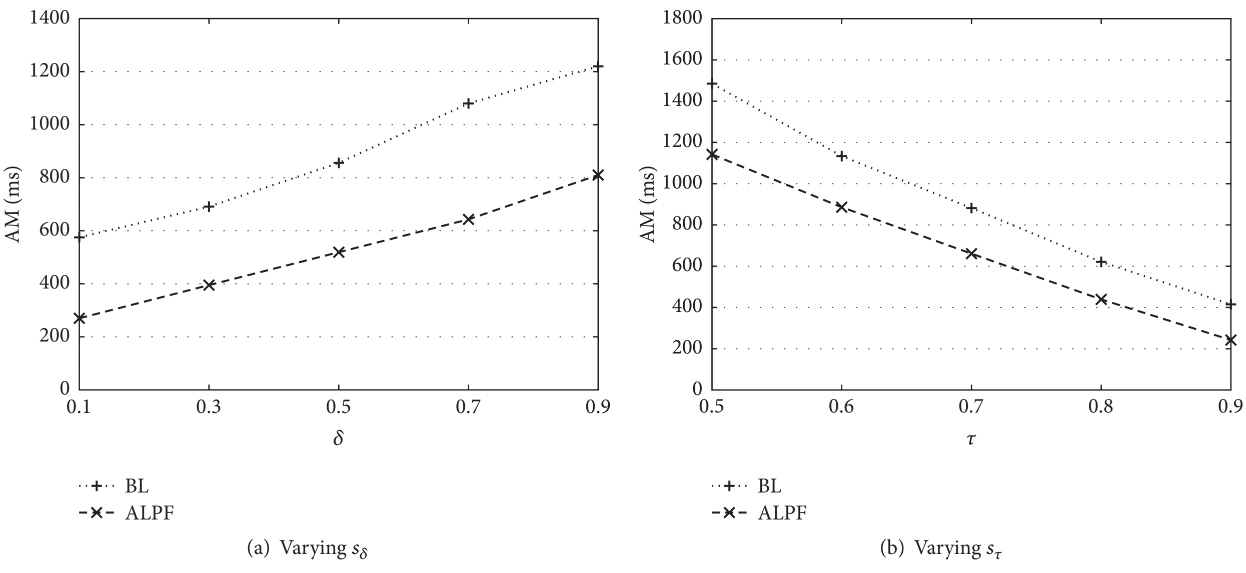


FIGURE 9: Evaluation of AM.

the commonly used hash function SHA1 [3]. Because SHA1 is collision-resistant and the hash value of the root of the TMR-tree is computed recursively from the messages that come within $\Delta t'$, which must include m, the recomputed root hash value of the TMR-tree cannot be verified against the signature, which can be detected by the subscriber s. Therefore, through our framework, subscribers can receive *sound* messages from the service provider. □

Proof of Completeness. Let m be a message satisfying the parameters δ and τ which is delivered to a subscriber s. For the recomputed hash value of the root of the TMR-tree to match the signature (i.e., the *soundness* is satisfied), there are the following two cases:

(i) The message m is included in the corresponding $\mathscr{VO}s$. In this case, the subscriber s can confirm whether this message is the result of using the returned spatial and textual information of m.

(ii) The message m is not included in the corresponding $\mathscr{VO}s$. In this case, it must be in the subtree rooted at R which is included in $\mathscr{VO}s$. However, the subscriber s cannot make sure that R does not satisfy δ and τ since if m is the result, R must satisfy δ and τ, which alarms the subscriber about potential violation of the *completeness*.

Therefore, through our framework, subscribers can receive *complete* messages from the service provider. □

5. Related Work

Our work is related to the location-aware publish/subscribe and authenticated query processing. Sections 5.1 and 5.2 retrospect the related work done in these areas.

5.1. Location-Aware Publish/Subscribe. Recently, location-aware publish/subscribe has attracted considerable attention. Most studies in this field can be categorized according to different evaluation methods of relevancy between subscriptions and messages [2, 4–8]. In particular, [4–7] use a spatial region to indicate the spatial information of each subscription and spatial overlap to evaluate spatial similarity and "**AND**", "**OR**" semantics or Boolean expressions to evaluate textual relevancy, while [2, 8] combine the textual relevancy and spatial similarity into a ranking function to quantify the relevancy between subscriptions and messages.

More specifically, regarding the first category, Chen et al. [4] study the problem of matching Boolean range continuous queries over a stream of incoming spatio-textual messages in real time. A Boolean range continuous query is to continually retrieve the spatio-textual messages arriving before the user-specified expiration time such that the retrieved spatio-textual messages satisfy the user's keywords which are connected by "**AND**" or "**OR**" semantics and are located in the query range. The authors present IQ-Tree, which is a hybrid index based on Quad-tree and inverted files. In [5], Li et al. study the location-aware publish/subscribe, which delivers a message to its corresponding subscribers having spatial overlap with the message and all the keywords in the subscriptions are contained in the message ("**AND**" semantic). They propose the R^t-tree, which extends the R-tree by selecting some representative keywords from subscriptions and adding them into R-tree nodes to enable textual pruning. Both matching algorithms of [4, 5] follow the filtering-and-refinement paradigm. More recently, although they study the same problem, Wang et al. [6] find that, in [4, 5], the spatial factor is always prioritized during the index construction regardless of the keyword distribution of the query set and the inverted indexing technique is not well-suited to textual filtering. Therefore, they utilize the keyword partition and space partition in one tree structure when constructing the index for queries based on expected matching cost. They compute the cost based on the number of queries associated with each partition and the probability of whether the partition is explored during message matching, instead of the complexity of filter and verification steps. Guo et al. [7] study filtering dynamic streams for continuous moving Boolean subscriptions. Different from previous works, it continuously monitors users' locations and sends nearby messages in real time and it allows users to specify their interests with Boolean expressions, which provides better flexibility and expressiveness in shaping an interest.

With respect to the second category, as introduced in Section 2.3, Hu et al. [2] study the parameterized location-aware publish/subscribe, which requires subscribers to specify parameters to enable personalized filtering. In [8], Chen et al. study top-k spatial-keyword publish/subscribe, which aims to continuously feed the user with new spatio-textual messages whose temporal spatial-keyword scores are ranked within the top-k. They use a Quad-tree to partition the whole space. Each subscription is assigned to a number of covering cells, forming a disjoint partition of the entire space and an inverted file ordered by subscription id is built to organize the subscriptions assigned to each cell.

5.2. Authenticated Query Processing. Authenticated query processing has been studied extensively. Most studies on query authentication are based on an $\mathscr{ADS}$, Merkle hash tree ($\mathscr{MHT}$) [1], as introduced in Section 2.3. The notion of the $\mathscr{MHT}$ is generalized to multiway trees and widely adapted to various index structures. Typical examples include the Merkle B-tree and its variant Embedded Merkle B-tree [9]. Following the concept of the $\mathscr{MHT}$, the authenticated query processing problem has also been studied for the relational data [9, 10], data streams [11–15], and textual search engines [16].

In the spatial databases domain, based on the $\mathscr{MHT}$, there are also many query authentication applications. Yang et al. [17] first introduce the query authentication problem to the domain of spatial data and study the authentication of spatial range queries. They propose an $\mathscr{ADS}$ called MR-tree, which combines the ideas of MB-tree [9] and R^*-tree [18]. Yiu et al. investigate how to efficiently authenticate moving kNN queries [19], moving range queries [20], and shortest-path queries [21]. More recently, Hu et al. [22] and Chen et al. [23] develop new schemes for range and top-k query authentication that preserve the location privacy of queried objects. Besides, Lin et al. [24] investigate the authentication of location-based skyline queries. A new $\mathscr{ADS}$ called MR-Sky-tree is proposed. Authentication of reverse k nearest neighbor query is studied by Li et al. in [25]. For the mixed data types, such as spatio-textual data, Su et al. [26] and Wu et al. [27] study the authentication problem for snapshot and moving top-k spatial-keyword queries, respectively. Yan et al. [28] explore the authentication problem in the area of spatio-textual similarity joins. Instead of only supporting the relational data as [10] does, the proposed authentication schemes in [28] can support spatial data. Zhang et al. [29] study the authentication of location-based top-k queries which ask for the POIs in a certain region and with the highest k ratings for an interested POI attribute.

Besides the $\mathscr{MHT}$, there are some other index structures which can be used to construct the $\mathscr{ADS}$, such as Voronoi diagram and prefix-tree. Hu et al. [30] propose a novel approach that authenticates spatial queries based on the neighborhood information derived from the Voronoi diagram. The problem of authenticating query results in data integration services is studied by Chen et al. in [31], which addresses multisource data authentication that can simultaneously support a wide range of query types. Based on the prefix-tree, they propose Homomorphic Secret Sharing Seal, which is to merge the authentication codes of nonresult values with a common prefix, thus allowing them to be verified as a whole.

6. Conclusion

In this paper we have studied the problem of authenticating messages in outsourced location-aware publish/subscribe services. We propose an authenticated location-aware publish/subscribe framework, including an $\mathscr{ADS}$ TMR-tree to organize the messages that come within $\Delta t'$, a filter-verification framework with inverted index pruning technique to efficiently deliver the messages to their relevant subscribers, and the methods of constructing the $\mathscr{VO}$ and authenticating the delivered messages at the subscribers side. Experimental results on a real-world dataset show that our framework achieves high performance.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 61502047 and the Coconstruction Program with the Beijing Municipal Commission of Education.

References

[1] R. C. Merkle, "A certified digital signature," in *Proceedings of the 9th Annual International Cryptology Conference on Advances in Cryptology*, vol. 435, pp. 218–238, Springer, August 1989.

[2] H. Hu, Y. Liu, G. Li, J. Feng, and K.-L. Tan, "A location-aware publish/subscribe framework for parameterized spatio-textual subscriptions," in *Proceedings of the 2015 31st IEEE International Conference on Data Engineering, ICDE 2015*, pp. 711–722, April 2015.

[3] Q. Dang, "Changes in Federal Information Processing Standard (FIPS) 180-4, Secure Hash Standard," *Cryptologia*, vol. 37, no. 1, pp. 69–73, 2013.

[4] L. Chen, G. Cong, and X. Cao, "An efficient query indexing mechanism for filtering geo-textual data," in *Proceedings of*

the 2013 ACM SIGMOD Conference on Management of Data, SIGMOD 2013, pp. 749–760, June 2013.

[5] G. Li, Y. Wang, T. Wang, and J. Feng, "Location-aware publish/subscribe," in *Proceedings of the the 19th ACM SIGKDD international conference*, p. 802, Chicago, Ill, USA, August 2013.

[6] X. Wang, Y. Zhang, W. Zhang, X. Lin, and W. Wang, "AP-Tree: Efficiently support continuous spatial-keyword queries over stream," in *Proceedings of the 2015 31st IEEE International Conference on Data Engineering, ICDE 2015*, pp. 1107–1118, April 2015.

[7] L. Guo, D. Zhang, G. Li, K.-L. Tan, and Z. Bao, "Location-aware pub/sub system: When continuous moving queries meet dynamic event streams," in *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2015*, pp. 843–857, June 2015.

[8] L. Chen, G. Cong, X. Cao, and K.-L. Tan, "Temporal Spatial-Keyword Top-k publish/subscribe," in *Proceedings of the 2015 31st IEEE International Conference on Data Engineering, ICDE 2015*, pp. 255–266, April 2015.

[9] F. Li, M. Hadjieleftheriou, G. Kollios, and L. Reyzin, "Dynamic authenticated index structures for outsourced databases," in *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, pp. 121–132, June 2006.

[10] Y. Yang, D. Papadias, S. Papadopoulos, and P. Kalnis, "Authenticated join processing in outsourced databases," in *Proceedings of the International Conference on Management of Data and 28th Symposium on Principles of Database Systems, SIGMOD-PODS'09*, pp. 5–17, July 2009.

[11] F. Li, K. Yi, M. Hadjieleftheriou, and G. Kollios, "Proof-infused streams: Enabling authentication of sliding window queries on streams," *VLDB*, pp. 147–158, 2007.

[12] S. Papadopoulos, Y. Yang, and D. Papadias, "Continuous authentication on data streams," *VLDB Journal*, pp. 135–146, 2007.

[13] S. Papadopoulos, Y. Yang, and D. Papadias, "Continuous authentication on relational streams," *VLDB Journal*, vol. 19, no. 2, pp. 161–180, 2010.

[14] S. Papadopoulos, A. Deligiannakis, G. Cormode, and M. Garofalakis, "Lightweight authentication of linear algebraic queries on data streams," in *Proceedings of the 2013 ACM SIGMOD Conference on Management of Data, SIGMOD 2013*, pp. 881–892, June 2013.

[15] S. Papadopoulos, G. Cormode, A. Deligiannakis, and M. Garofalakis, "Lightweight query authentication on streams," *ACM Transactions on Database Systems*, vol. 39, no. 4, article 30, 45 pages, 2014.

[16] H. Pang and K. Mouratidis, "Authenticating the query results of text search engines," *VLDB*, pp. 126–137, 2008.

[17] Y. Yang, S. Papadopoulos, D. Papadias, and G. Kollios, "Spatial outsourcing for location-based services," in *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering, ICDE'08*, pp. 1082–1091, April 2008.

[18] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger, "R*-an efficient and robust access method for points and rectangles," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 322–331, May 1990.

[19] M. L. Yiu, E. Lo, and D. Yung, "Authentication of moving kNN queries," in *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering, ICDE 2011*, pp. 565–576, April 2011.

[20] D. Yung, E. Lo, and M. L. Yiu, "Authentication of moving range queries," in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM 2012*, pp. 1372–1381, November 2012.

[21] M. L. Yiu, Y. Lin, and K. Mouratidis, "Efficient verification of shortest path search via authenticated hints," in *Proceedings of the 26th IEEE International Conference on Data Engineering, ICDE 2010*, pp. 237–248, March 2010.

[22] H. Hu, J. Xu, Q. Chen, and Z. Yang, "Authenticating location-based services without compromising location privacy," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, SIGMOD '12*, pp. 301–312, May 2012.

[23] Q. Chen, H. Hu, and J. Xu, "Authenticating top-k queries in location-based services with confidentiality," *VLDB*, vol. 7, no. 1, pp. 49–60, 2013.

[24] X. Lin, J. Xu, and H. Hu, "Authentication of location-based skyline queries," in *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM'11*, pp. 1583–1588, October 2011.

[25] G. Li, C. Luo, and J. Li, "Authentication of reverse k nearest neighbor query," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9049, pp. 625–640, 2015.

[26] S. Su, H. Yan, X. Cheng, P. Tang, P. Xu, and J. Xu, "Authentication of top-k spatial keyword queries in outsourced databases," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9049, pp. 567–588, 2015.

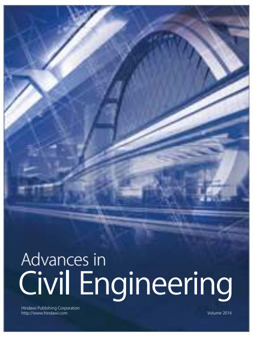
[27] D. Wu, B. Choi, J. Xu, and C. S. Jensen, "Authentication of Moving Top-k Spatial Keyword Queries," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 4, pp. 922–935, 2015.

[28] H. Yan, X. Cheng, S. Su, Q. Zhang, and J. Xu, "Authenticated spatio-textual similarity joins in untrusted cloud environments," *ICPADS*, pp. 685–694, 2016.

[29] R. Zhang, Y. Zhang, and C. Zhang, "Secure top-k query processing via untrusted location-based service providers," in *Proceedings of the IEEE Conference on Computer Communications, INFOCOM 2012*, pp. 1170–1178, March 2012.

[30] L. Hu, W. Ku, S. Bakiras, and C. Shahabi, "Spatial query integrity with voronoi neighbors," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 4, pp. 863–876, 2013.

[31] Q. Chen, H. Hu, and J. Xu, "Authenticated online data integration services," in *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2015*, pp. 167–181, June 2015.



Hindawi

Submit your manuscripts at
https://www.hindawi.com