Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2013, Article ID 302147, 8 pages
http://dx.doi.org/10.1155/2013/302147

Research Article

SSE: A Secure Searchable Encryption Scheme for Urban Sensing and Querying

Mi Wen, Jingsheng Lei, and Zhongqin Bi

School of Computer Engineering, Shanghai University of Electronic Power, Shanghai 200090, China

Correspondence should be addressed to Jingsheng Lei; jshlei@126.com

Received 4 July 2013; Accepted 27 October 2013

Academic Editor: Yunhuai Liu

With the distributed sensors that are deployed to monitor the urban environment in smart cities, the sensed data are overwhelming and beyond the scope of the wireless sensor networks (WSNs) capability. Due to the communication range limits of the sensors, most of these data will be outsourced and stored on some untrusted servers. Thus, how to maintain the data confidentiality and integrity, as well as source authentication and data query privacy of the outsourced data, is a challenging problem. In this paper, we propose a secure searchable encryption scheme, named SSE, for urban sensing and querying to address the problem. Specifically, our SSE constructs a secure hidden vector encryption-(HVE-) based rang query predicate. The sensed data can be stored on an untrusted server in encrypted form. A requester can obtain the correct ciphertexts when his authorized range query matches the HVE-based encryption predicate. With the help of the base station, the ciphertexts can be decrypted and data integrity can be verified; then, the requester can obtain the correct original data. Security analysis demonstrates that; in the SSE, only the authorized requesters can obtain the query results, while the data confidentiality and integrity and source authentication are also preserved.

1. Introduction

Nowadays, with the continuous expansion of urban populations, there are increasing needs in many aspects related to urban living, such as environment monitoring, and public safety supervision [1]. Especially, a city can be defined as "smart" when modern information and communication infrastructures (ICI) communication can automatically control the urban living in a secure means. Wireless sensor networks (WSNs) increasingly become viable solutions to urban sensing and querying for smart cities. Widely distributed sensors monitor the urban environment in real time and collect data for intelligent decision making by multihop wireless transmission, as shown in Figure 1, which can facilitate various services and improve the quality of urban living [2]. However, deploying sensors without security in mind has often proved to be dangerous in hostile environments [3]. Especially, when there is terrorism, the detection of suspicious activities should be reported to the proper authorities in a secure approach.

In the worst case scenario, a single corrupted message can lead to the ICI systems in a total tizzy. For example, if a fake traffic accident information is broadcasted in rush hours, the ICI systems maybe make some misleading decisions to the drivers; thus, traffic jams will be caused in that city. This kind of fake data injection attacks not only prevent the authorities from collecting correct messages but also exhaust the energy of the forwarders [4]. Consequently, it is essential to prevent the malicious node from impersonating good nodes for spreading misleading information intentionally. Hence, sensitive monitoring data should be transmitted in encrypted form to achieve data confidentiality, integrity, and authentication between communicating parties. Furthermore, with the distributed sensors, the data collections are going to be several millions a day. In order to save energy and relieve the burden of data storage and maintenance [5], data outsourcing is the best way to let sensors store their sensed data on some untrusted servers (cloud or third party provided servers) in encrypted form and execute computation and queries using server's computational capabilities. In addition, universal data access with independent geographical locations, avoidance of capital expenditure on hardware, software, and personnel maintenances will make the smart city's ICI system more reliable and efficient.

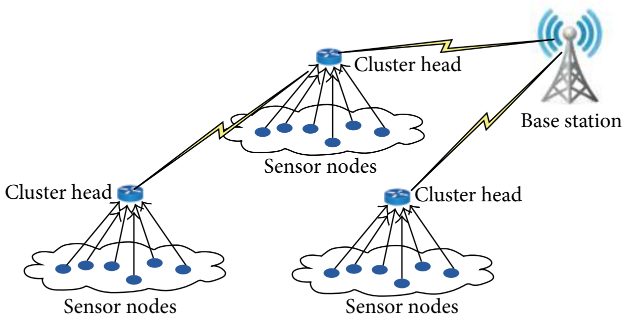

Figure 1: The conceptional model of wireless sensor network.

Naturally, storing data in encrypted form on untrusted server can maintain data confidentiality for the stored data, as long as a secret key corresponding to a certain encryption is not given out to untrusted servers [6]. However, if these data are needed later for matters such as: calculation, analysis, or for intelligent decision-making, searchers (e.g., decision maker) are endowed with the task of querying these data, which can help them to gain organization-wide situational awareness. In this case, querying encrypted data is a major logistic problem. The server is anticipated to be able to evaluate various query predicates against the data without having to decrypt it [7]. Especially, when the query conditions are composed of many conjunctive range requirements (e.g., date ranges and geographic regions, etc.), rang query over encrypted data is much more difficult.

In this paper, we propose a secure searchable encryption scheme (SSE) for urban sensing and querying. The SSE addresses the data confidentiality, integrity, and source authentication by using a secure hidden vector encryption-(HVE-) based range query predicate. The main contributions of this paper are twofold.

(i) Firstly, we propose a secure HVE-base range query predicate. Specifically, a sensor node encrypts its data by using the shared key with the BS. Then, it hides the searchable attributes and the ciphertext into the HVE-base range query predicate. Only when the two vectors associated with the encryption attributes and range query are component-wise equal, the ciphertext can be recovered.

(ii) Secondly, we analyze the security strengths of the SSE. The analysis results demonstrate that, compared with existing schemes, the SSE is the only one which can achieve data confidentiality and integrity, as well as source authentication and data query privacy.

The remainder of this paper is organized as follows. In Section 2, we discuss the related works. In Section 3, we introduce our system model and security requirements. Then, in Section 4, we review some preliminaries. In Section 5, we construct a secure HVE-based range query predicate. In Section 6, we present our SSE scheme, followed by its security analysis in Section 7. Finally, we conclude this paper in Section 8.

2. Related Works

2.1. Security in Sensor Network. Security in sensor network has been widely studied. Many security proposals for WSNs have focused on efficient key management in WSNs. Some are based on the symmetric cryptosystems. For instance, Perrig et al. [8] propose SPINS, a suite of efficient symmetric key-based security building blocks. Eschenauer and Gligor [9] look at random key predistribution schemes, which open the way to a large number of follow-up works [10]. Zhu et al. [11] proposed LEAP, a rather efficient scheme based on local distribution of secret keys among neighboring nodes. These strategies, however, do not provide perfect resilience, as after a certain percentage of nodes have been compromised, the whole network can be compromised as well. Others have employed public key cryptography (PKC) to adjust conventional algorithms (e.g., RSA [12]) to sensor nodes or to employ more efficient techniques (e.g., NTRU [3], ECC [13]) in this resource constrained environment. However, they all use interactive protocols and therefore nodes are required to exchange messages to agree on keys. Moreover, they are not suitable for data outsourcing and data query. He and Li schedule the sequence of query processing to achieve the optimized overall energy efficiency by fully utilizing the implications among sensor networks [14]. It also cannot be applied to query over encrypted data.

2.2. Range Query over Encrypted Data. The problem of range query over encrypted data is a hot issue in both cryptography and database communities [6, 15–17]. There are essentially four categories of solutions that have been developed for range queries. One general approach is to ensure that order amongst plaintext data is preserved in the ciphertext domain by using order-preserving encryption-based (OPE) techniques [15]. This allows direct translation of range predicates from the original domain to the domain of the ciphertext. As discussed in [18], the coupling distribution of plaintext and ciphertext domains might be exploited by attackers to guess the scope of the corresponding plaintext for a ciphertext. Another bucketization-based (Buck) technique [6] is to use distributional properties of the dataset to partition and index them for efficient querying while trying to keep the information disclosure to a minimum. Queries are evaluated in an approximate manner where the returned set of records may contain some false positives. These records will expose unrelated data's confidentiality and increase the computational and communication overhead of the system.

Thirdly, those that use some specialized data structure for range query evaluation while trying to preserve notions of semantic security of the encrypted data [17], such as B+tree [19]. In order to hide the access patterns, they run fake queries. In addition, they buffer the first few levels of the tree on the client side. Finally, they shuffle from time to time. The last approach is using a specialized type of predicate encryption, the HVE-based approach [20], where two vectors over attributes are associated with a ciphertext and a token, respectively. Under predicate translator, the ciphertext matches the token if and only if the two vectors are component-wise equal. Although most of the existing

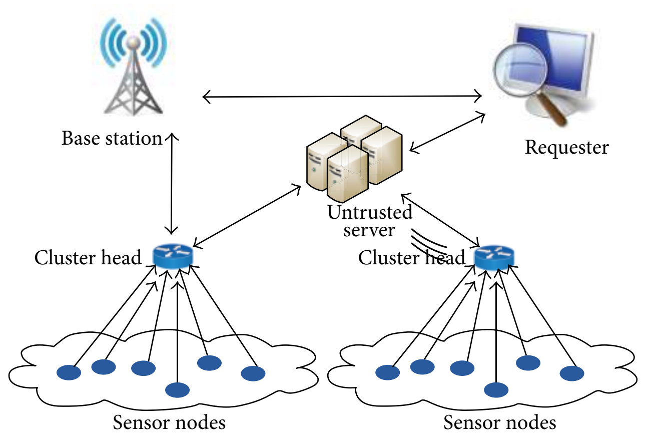

Figure 2: System model of SSE.

range query approaches can maintain part of the data confidentiality, all of them are not secure enough to keep the untrusted server from peering the original data when the requester's query is successful. Therefore, the goal of this paper is to propose a secure range query predicate to support data confidentiality and integrity, as well as source authentication and data query privacy.

3. System Model and Security Requirements

In this section, we formalize the system model and identify the security requirements and our design goals.

3.1. System Model. In this paper, we consider a large-scale WSNs consisting of a database server (DB), a base station (BS), some cluster heads (CHs), and numerous sensor nodes (SNs) which are grouped into clusters. The clusters can be formed based on various criteria such as capabilities, location, and communication range [21]. Each CH controls a cluster and is responsible for assigning public group information and distributing some command to its cluster members. Each CH is assumed to be reachable to all sensors in its cluster, either directly or by multihop. We also assume that each node is innocent before deployment and cannot be compromised during the first several minutes after deployment since compromising a node takes some time. The system model is depicted in Figure 2. We require that the sensed data from each SN can be stored in a DB by its corresponding CH. Due to the large distances between cluster and the DB, communication from SN to DB can only be performed via CH. This assumption is generally true, since the communication range of each SN is limited. The DB can be cloud servers or other local third party servers; they are honest but curious; sometimes it is untrusted.

All the SNs encrypt their data by using the shared keys with the BS and hide their searchable attributes by using the HVE-based encryption predicate. When a requester S got an authorized query tokens from the BS, he can get the corresponding corrected querying results from DB. Next, with the help of the BS, S can decrypt and authenticate the original data. Thus, the BS is trustable.

3.2. Security Requirements. We identify the security requirements for our SSE. In our security model, the BS operates as a trusted authority (TA). The sensors $\mathbb{N} = \{N_1, N_2, \ldots, N_v\}$ are honest as well. However, there exists an adversary $\mathscr{A}$ in the system to eavesdrop and invade the database on DB to steal the individual SN's reports. In addition, $\mathscr{A}$ can also launch some active attacks to threaten the data confidentiality and integrity. Therefore, in order to prevent $\mathscr{A}$ from learning the SNs' data and to detect $\mathscr{A}$'s malicious actions, the following security requirements should be satisfied in range query applications for smart cities.

(i) *Data Confidentiality*. Sensed data should not be disclosed by unauthorized parties and it is the most important issue in mission critical applications. Moreover, in many applications, SNs transmit highly sensitive data, for example, terrorism supervision data, and therefore it is extremely important to build secure channels between SNs and the authorities. The standard approach for keeping sensitive data secret is to encrypt the data with a secret key that only intended authorized receivers possess, hence achieving confidentiality.

(ii) *Data Integrity*. A malicious node may just corrupt messages to prevent network from functioning properly. Thus, sensitive data needs to be protected from being altered. Message authentication code (MAC) is a widely used approach to guarantee that the message being transferred is never corrupted. Therefore, data integrity can be achieved.

(iii) *Source Authentication*. Since wireless sensor networks use a shared wireless medium, without source authentication, an adversary could masquerade a node, thus gaining unauthorized access to resource and sensitive information and interfering with the operation of other nodes. Moreover, a compromised node may send data to its CH under several fake identities so that the decision-making is misled.

(iv) *Data Query Privacy.* Since SN's sensed data are stored on the untrusted DB in an encrypted form, generally, the server cannot know the original data. In this case, when a requester's query vectors in the query tokens match the encryption vectors, the original data can be exposed to the DB. This is also the security vulnerability of the data privacy. Therefore, double encryption is needed to ensure data privacy when range query succeeds.

4. Preliminaries

In this section, we will briefly describe the basic definition and properties of bilinear pairings, polynomial-based key predistribution, and HVE-based comparison query predicate.

4.1. Bilinear Pairing. Bilinear pairing is an important cryptographic primitive [22]. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic multiplication groups of prime order q. Let a and b be elements of Z_q^*. We assume that the discrete logarithm problem (DLP) in both $\mathbb{G}_1$ and $\mathbb{G}_2$ is hard. g is a generator of $\mathbb{G}_1$. A bilinear pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ with the following properties:

(1) bilinear: $e(g^a, h^b) = e(g, h)^{ab}$ for any $(g, h) \in \mathbb{G}_1^2$;

(2) nondegenerate: $e(g, h) \neq 1_{\mathbb{G}_2}$ whenever $g, h \neq 1_{\mathbb{G}_1}$;

(3) computable: there is an efficient algorithm to compute $e(g, h) \in \mathbb{G}_2$ for all $(g, h) \in \mathbb{G}_1^2$.

Definition 1. A bilinear parameter generator $\mathcal{G}en$ is a probabilistic algorithm that takes a security parameter κ as input and outputs a 5-tuple $(q, g, \mathbb{G}_1, \mathbb{G}_2, e)$.

4.2. Polynomial-Based Key Predistribution. In this section, we briefly review the polynomial-based key predistribution approach [23]. To predistribute shared keys, the TA randomly generates a bivariate t-degree polynomial $h(x, y) = \sum_{i,j=0}^{t} a_{ij} x_i y_j$ over a finite field F_q, where q is a large prime number. The polynomial has the property of $h(x, y) = h(y, x)$. For example, if two sensor nodes i and j need to establish a shared symmetric key, the KDC computes a polynomial share of $h(x, y)$, that is, $h(i, y)$ to the node i. Then, node i can compute the common key $h(i, j)$ by evaluating $h(i, y)$ at point j, and node j can compute the shared symmetric key $h(j, i) = h(i, j)$ by evaluating $h(j, y)$ at point i.

4.3. HVE-Based Comparison Query Predicate. HVE [16] is a type of predicate encryption where two vectors over attributes are associated with a ciphertext and a token, respectively. The ciphertext matches the token if and only if the two vectors are component-wise equal. There are two character sets $\sum$ and $\sum_* = \sum \cup \{*\}$ in the setting of HVE. Here, $\sum$ is an arbitrary set of attributes. We assume $\sum = \mathbb{Z}_q$; $*$ is a special symbol denoting a wildcard component, which means that the component related to $*$ is not involved with any attribute. HVE mainly consists of four phases: key generation, data encryption, token generation, and data query.

In key generation phase, a TA distributes the public/private key pair (PK, SK) to a receiver.

Then, in the data encryption phase, a SN chooses a vector $\mathbf{x} = (x_1, \dots, x_l) \in \sum^l$ to characterize its data. If we map the ith component $x_i \in \mathbf{x}$ to a number of its domain $j \in \{1, \dots, n\}$ as in [20], the value of x_i is one of the number in set $\{1, \dots, n\}$. The SN builds an encryption vector $\sigma(\mathbf{x}) = (\sigma_{i,j}) \in \{0, 1\}^{nl}$ for $\mathbf{x} = (x_1, \dots, x_l) \in \{1, \dots, n\}^l$, as follows:

$$\sigma_{i,j} = \begin{cases} 1, & \text{if } x_i \geq j, \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

where $i \in \{1, \dots, l\}$ and $j \in \{1, \dots, n\}$. For example, $l = 3$, $n = 5$ and let $\mathbf{x} = (1, 3, 2)$. Thus, $\mathbf{x} = (x_1, \dots, x_l) \in \{1, \dots, n\}^l = \{1, 2, 3, 4, 5\}^3$ and the corresponding encryption vector $\sigma(\mathbf{x}) = (10000, 11100, 11000)$. Then, SN's data m should be encrypted into a ciphertext CT under the encryption vector $\sigma(\mathbf{x})$ and the receiver's public key.

Next, in the token generation phase, the requester chooses a vector $\mathbf{w} = (w_1, \dots, w_l) \in (\sum_*)^l$ to represent his query bound and builds a query vector $\sigma^*(\mathbf{w}) = (\sigma_{i,j}^*) \in \{0, 1, *\}^{nl}$ for $\mathbf{w} = (w_1, \dots, w_l) \in \{1, \dots, n\}^l$ as follows:

$$\sigma_{i,j}^* = \begin{cases} 1, & \text{if } w_i = j, \\ *, & \text{otherwise.} \end{cases} \tag{2}$$

Similarly, we assume that $l = 3$, $n = 5$, and $\mathbf{w} = (w_1, \dots, w_l) \in \{1, \dots, n\}^l = \{1, 2, 3, 4, 5\}^3$. For example, if the receiver's query condition is $P = (x_1 \geq 1) \wedge (x_2 \geq 3) \wedge (x_3 \geq 1)$; that is, $\mathbf{w} = (1, 3, 1)$. Thus, the query vector $\sigma^*(\mathbf{w}) = (1 * * * *, * * 1 * *, 1 * * * *)$. Note that, the number of the elements in $\sigma^*(\mathbf{w})$ is nl. Based on the query vector $\sigma^*(\mathbf{w})$, a query token T_P is generated. The receiver sends T_P to the server.

In the data query phase, let $s(\sigma^*(w))$ denote the set of all indexes k which satisfies $\sigma_k^* \neq *$, where $k \in \{1, \dots, nl\}$. Let $P_{\sigma^*(\mathbf{w})}(\sigma(\mathbf{x}))$ be the following comparison predicate:

$$P_{\sigma^*(\mathbf{w})}(\sigma(\mathbf{x})) = \begin{cases} 1, & \text{if } \forall i \in s(\sigma^*(w)),\ \sigma^*(w_i) = \sigma(x_i), \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Finally, the server can disclose the data m if the comparison predicate $P_{\sigma^*(\mathbf{w})}(\sigma(\mathbf{x})) = 1$.

5. Construction of Secure HVE-Based Range Query Encryption Predicate

In this section, we modified the HVE scheme in [7] to support data query privacy. To protect the original data from exposing to the untrusted server even when the range query succeeds, double encryption is needed. Firstly, sensed data m can be encrypted into a ciphertext C by using its shared secret key with the BS. Then, to enable range query on the untrusted DB, the ciphertext C can be encrypted into a searchable cipher CT by using the HVE-based range query encryption predicate. Consequently, the DB can only get the ciphertext C when the requester's query tokes match the encryption vectors. Finally, the requester can get the original data with the help of the BS.

Three entities: sender, server, and requester are involved in this section. It mainly consists of the following five phases: key generation phase, data encryption phase, token generation phase, data query phase, and data decryption phase.

(1) In key generation phase, a TA also distributes the public/private key pair (PK, SK) to a receiver. We extend the comparison predicate to support range query predicate. Specifically, we can achieve the opposite semantics of the above comparison query in (1), that is, $x_i \leq j$, by constituting the vectors $\sigma(\mathbf{x})$ in a reverse manner as follows:

$$\sigma_{i,j} = \begin{cases} 1, & \text{if } x_i \leq j, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Thus, the encryption predicate can support range queries, such as $a \leq x_i \leq b$.

(2) In the data encryption phase, the SN should define two encryption vectors: $\sigma_g(\mathbf{x})$ and $\sigma_s(\mathbf{x})$ according to (1) and (4) when $x_i \geq j$ and $x_i \leq j$, respectively. The receiver can get the correct data if and only if both conditions $x_i \geq a$ and $x_i \leq b$ hold. Suppose that, if the encrypted data in HVE is M, the SN $N_i \in \mathbb{N}$ should encrypt M into a ciphertext C. Then, N_i split C into different parts by using secret sharing: randomly chooses a polynomial $f(x) = \alpha_0 + \alpha_1 x + \cdots + \alpha_t x^{t-1}$, where $\alpha_0 = C$ and α_i are randomly coefficients. In our scheme, C only needs to be divided into two directions of the range query; thus, $t = 2$. Then, N_i chooses two random integers ξ, π and computes two data shares $f(\xi)$ and $f(\pi)$; that is, C is divided into two parts: $f_0 = f(\xi)$ and $f_1 = f(\pi)$. N_i encrypts f_0 and f_1 under vectors $\sigma_g(\mathbf{x})$ and $\sigma_s(\mathbf{x})$, respectively. For example, if we assume $l = 3$, $n = 5$, and $\mathbf{x} = (2,3,4)$, f_0 is encrypted under the vector $\sigma_g(\mathbf{x}) = (11000, 11100, 11110)$, as shown in Table 1. N_i outputs CT_0. Also, f_1 is encrypted under the vector $\sigma_s(\mathbf{x}) = (01111, 00111, 00011)$, as shown in Table 2. Similarly, N_i outputs CT_1. Thus, ciphertext C is encrypted into ciphertexts CT_0 and CT_1.

(3) In the token generation phase, the requester's range query is defined with two vectors: $\sigma_g^*(\mathbf{w})$ and $\sigma_s^*(\mathbf{w})$ when $x_i = a$ and $x_i = b$, respectively. Let $s(\sigma_g^*(w))$ be the set of all indexes k which satisfies $\sigma_g^*(w_k) \neq *$, and let $s(\sigma_s^*(w))$ be the sets of all indexes k' which satisfies $\sigma_s^*(w_{k'}) \neq *$. Here, $k, k' \in (1, \ldots, nl)$. Finally, in the data query phase, the server checks two comparison predicates $P_{\sigma_g^*(\mathbf{w})}(\sigma_g(\mathbf{x}))$ and $P_{\sigma_s^*(\mathbf{w})}(\sigma_s(\mathbf{x}))$, which are generated according to (4). The server can obtain f_0 if $\sigma_g(x_k)$ and $\sigma_g^*(w_k)$ are equal for all $k \in s(\sigma^*(w_g))$; that is, $P_{\sigma_g^*(\mathbf{w})}(\sigma_g(\mathbf{x})) = 1$. Similarly, the server can obtain f_1 if $\sigma_s(x_{k'})$ and $\sigma_s^*(w_{k'})$ are equal for all $k' \in s(\sigma_s^*(w))$; that is, $P_{\sigma_s^*(\mathbf{w})}(\sigma_s(\mathbf{x})) = 1$. The range query predicate can be denoted as follows:

$$P_{(\sigma_g^*(\mathbf{w}),\sigma_s^*(\mathbf{w}))}\left(\sigma_g(\mathbf{x}), \sigma_s(\mathbf{x})\right) = \begin{cases} 1, & \text{if } P_{\sigma_g^*(\mathbf{w})}\left(\sigma_g(\mathbf{x})\right) = 1,\ P_{\sigma_s^*(\mathbf{w})}\left(\sigma_s(\mathbf{x})\right) = 1, \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

For instance, a range query $P = (2 \leq x_1 \leq 3) \wedge (2 \leq x_2 \leq 4) \wedge (3 \leq x_3 \leq 5)$ can be divided into two parts: $P_g = (x_1 \geq 2) \wedge (x_2 \geq 2) \wedge (x_3 \geq 3)$ and $P_s = (x_1 \leq 3) \wedge (x_2 \leq 4) \wedge (x_3 \leq 5)$. Thus, $w_g = (2,2,3)$, $w_s = (3,4,5)$. Two query tokens T_{Pg} and T_{Ps} are computed under the vectors $\sigma_g^*(w) = (*1***, *1***, **1**)$ and $\sigma_s^*(w) = (**1**, ***1*, ****1)$, respectively.

(4) In the data query phase, when the query tokens T_{Pg} and T_{Ps} are deposited to the DB. As shown in Table 1, if the corresponding position of vectors $\sigma_g(\mathbf{x})$ and $\sigma_g^*(\mathbf{w})$ are equal for all $k \in s(\sigma_g^*(w))$, the ciphered f_0 can be recovered by using CT_0 and T_{Pg}. Similarly, as shown in Table 2, if the corresponding position of vectors $\sigma_s(\mathbf{x})$ and $\sigma_s^*(w)$ are equal for all $k' \in s(\sigma_s^*(w))$, f_1 will be recovered by using CT_1 and T_{Ps}. Then, the ciphertext C can be computed from f_0 and f_1 when $P_{(\sigma_g^*(\mathbf{w}),\sigma_s^*(\mathbf{w}))}(\sigma_g(\mathbf{x}), \sigma_s(\mathbf{x})) = 1$.

(5) In the data decryption phase, with the help of the BS, the ciphertext C can be decrypted. Finally, the BS distributes the data M to the requester in a secure channel.

6. Proposed SSE Scheme

In this section, we describe the details of our SSE scheme. There are also five phases: key generation phase, data encryption phase, token generation phase, data query phase, and data decryption and verification phase.

6.1. Key Generation Phase. For our scheme, we assume that there is a TA (actually is the BS) which can bootstrap the whole system. Specifically, in this system initialization phase, the TA generates a bivariate t-degree polynomial $h(x,y) = \sum_{i,j=0}^{t} \alpha_{ij} x_i y_j$. TA computes two polynomial share: $h(x, \mathrm{BS})$ for BS itself and $h(i, y)$ for the node N_i. $h(N_i, y)$ will be preloaded in the S's memory. A symmetric encryption algorithm $\mathrm{Enc}(\cdot)$ is used to encrypt the data m_i, for example, AES. Given the security parameter 1^k, the TA first generates $(q, g, \mathbb{G}_1, \mathbb{G}_2, e)$ by running $\mathscr{G}en(1^k)$. TA selects some random elements $g, g_1, g_2, (h_1, u_1, \psi_1), \ldots, (h_{nl}, u_{nl}, \psi_{nl}) \in \mathbb{G}_1$. It also picks random numbers $y_1, y_2, v_1, \ldots, v_{nl}, t_1, \ldots, t_{nl} \in \mathbb{Z}_p$. Then, it computes $Y_1 = g^{y_1}, Y_2 = g^{y_2}, v_k = g^{v_k} \in \mathbb{G}_1$ for $(k = 1, \ldots, nl)$. In addition, it computes $A = e(g_1, Y_1) \cdot e(g_2, Y_2) \in \mathbb{G}_2$. Then, TA distributes the public/private key pair (PK, SK) to the BS as follows:

$$\mathrm{PK} = \left(g, Y_1, Y_2, (h_1, u_1, \psi_1, V_1, T_1), \ldots, (h_{nl}, u_{nl}, \psi_l, V_{nl}, T_{nl})\right)$$
$$\mathrm{SK} = \left(g_1, g_2, y_1, y_2, v_1, \ldots, v_{nl}, t_1, \ldots, t_{nl}\right). \quad (6)$$

6.2. Data Encryption Phase. If $N_i\,(1 \leq i \leq n)$ wants to report a data m_i to the BS, each data has l searchable attributes; N_i chooses a vector $\mathbf{x}_i = (x_{i1}, \ldots, x_{il}) \in \sum^l$ to characterize its data m_i in different dimensions. N_i computes a message authentication code (MAC) for m_i by using its secret key with the BS as follows: N_i inputs BS's identity into its polynomial share $h(N_i, y)$ and obtains a shared secret key $k_i = h(N_i, \mathrm{BS})$ with the BS. N_i encrypts his data m_i by using k_i as $C_i = En_{k_i}(m_i)$. Then, the BS computes an $\mathrm{MAC}_i = HMAC_{k_i}(m_i)$ for authenticating data m_i's integrity. Also, N_i computes another $\mathrm{MAC}_i' = HMAC_{k_i}(N_i)$ to authenticate his identity N_i.

In order to support range query "$a_j \leq x_{ij} \leq b_j$" $(1 \leq j \leq l)$, C_i should be divided into two parts as we described in Section 5. Only when both conditions "$x_{ij} \geq a_j$" and "$x_{ij} \leq b_j$" are satisfied, C_i can be recovered. Therefore, the N_i divides each C_i into two parts: f_{i0} and f_{i1}. f_{i0} is encrypted

Table 1: The larger condition vectors.

$\sigma_g(\mathbf{x})$	j 1	2	3	4	5	$\sigma_g^*(\mathbf{w})$	j 1	2	3	4	5
$\sigma_g(x_1)$	1	1	0	0	0	$\sigma_g^*(w_1)$	$*$	1	$*$	$*$	$*$
$\sigma_g(x_2)$	1	1	1	0	0	$\sigma_g^*(w_2)$	$*$	1	$*$	$*$	$*$
$\sigma_g(x_3)$	1	1	1	1	0	$\sigma_g^*(w_3)$	$*$	$*$	1	$*$	$*$

Table 2: The less condition vectors.

$\sigma_s(\mathbf{x})$	j 1	2	3	4	5	$\sigma_s^*(\mathbf{w})$	j 1	2	3	4	5
$\sigma_s(x_1)$	0	1	1	1	1	$\sigma_s^*(w_1)$	$*$	$*$	1	$*$	$*$
$\sigma_s(x_2)$	0	0	1	1	1	$\sigma_s^*(w_2)$	$*$	$*$	$*$	1	$*$
$\sigma_s(x_3)$	0	0	0	1	1	$\sigma_s^*(w_3)$	$*$	$*$	$*$	$*$	1

by using the encryption vector $\sigma_g(\mathbf{x}_i)$; f_{i1} is encrypted by using the encryption vector $\sigma_s(\mathbf{x}_i)$. Thus, the BS can recover C_i only when both encryption vectors are satisfied with the corresponding query vectors in the range query tokens. The encryption details are as follows:

(1) N_i maps $\mathbf{x}_i$ to an encryption vector $\sigma_g(\mathbf{x}_i)$ as (1). It selects two random numbers $r_{i1}, r_{i2} \in Z_p$, and computes some tags for the partial data f_{i0} by using the mapped vector $\sigma_g(\mathbf{x}_i)$ as

$$\begin{aligned} C_{i10} &= Y_1^{r_{i1}}, \qquad C_{i20} = Y_2^{r_{i2}}, \\ C_{i3,10} &= \left(h_1 u_1^{\sigma_g(x_{i1})}\right)^{r_1} V_1^{r_{i2}}, \\ &\ldots, \\ C_{i3,nl0} &= \left(h_{nl} u_{nl}^{\sigma_g(x_{inl})}\right)^{r_{i1}} V_{nl}^{r_{i2}}, \\ C_{i4,10} &= \psi_1^{r_{i1}} T_1^{r_{i2}}, \\ &\ldots, \\ C_{i4,nl0} &= \psi_{nl}^{r_{i1}} T_{nl}^{r_{i2}}, \\ C_{i50} &= g^{r_{i2}}, \\ C_{i60} &= A^{r_{i1}} C. \end{aligned} \tag{7}$$

Let $CT_{i0} = (C_{i10}, C_{i20}, C_{i3,10}, \ldots, C_{i3,nl0}, C_{i4,10}, \ldots, C_{i4,nl0}, C_{i50}, C_{i60})$. Similarly, N_i maps $\mathbf{x}_i$ to an encryption vector $\sigma_s(\mathbf{x}_i)$ as (4). N_i can encrypt the partial data m_{i1} into ciphertext CT_{i1} by using the encryption vector $\sigma_s(\mathbf{x}_i)$. Then, N_i deposits CT_{i0} and CT_{i1} to its cluster head CH_i as

$$N_i \longrightarrow \mathrm{CH}_i : \left\{N_i, CT_{i0}, CT_{i1}, ts_i, \mathrm{MAC}_i, \mathrm{MAC}_i'\right\}. \tag{8}$$

(2) The CH also adds the cluster number cl_i to the ciphertexts and transmits them to the BS

$$\mathrm{CH}_i \longrightarrow \mathrm{DB} : \left\{N_i, CT_{i0}, CT_{i1}, ts_i, \mathrm{MAC}_i, \mathrm{MAC}_i', cl_i\right\}. \tag{9}$$

6.3. Token Generation Phase. In this phase, firstly, the BS will authenticate the requester S's identity by using any identity-based mechanism [24]. Then, if S is a valid data user, the BS will help requester to transfer its range query into query tokens. If the range query is $P = (a_1 \le x_1 \le b_1) \wedge (a_2 \le x_2 \le b_2) \cdots \wedge (a_l \le x_l \le b_l)$. For each querying interval

$$S \longrightarrow \mathrm{BS} : \{P\}. \tag{10}$$

Then, BS divides P into two parts: $P_g = (x_1 \ge a_1) \wedge (x_2 \ge a_2) \cdots \wedge (x_l \ge a_l)$ and $P_s = (x_1 \le b_1) \wedge (x_2 \le b_2) \cdots \wedge (x_1 \le b_l)$. Let $w_g = (a_1, \ldots, a_l)$ and $w_s = (b_1, \ldots, b_l)$.

The BS generates a predicate vector $\sigma_g^*(\mathbf{w}) \in (\sum_*)^{nl}$ to represent P_g. The wildcard $*$ in the vector $\sigma_g^*(\mathbf{w})$ means that he does not care about the attributes related to $*$. Let $s(\sigma_g^*(w))$ be the set of all indexes k such that $\sigma_g^*(w) \ne *$. Then, a token T_{Pg} will be computed under the vector $\sigma_g^*(\mathbf{w})$ as follows.

(1) Select a random $\alpha, \beta \in Z_p$ and generate $\lambda_j, k_j, \gamma_j, \tau_j \in Z_p$ such that $\lambda_j y_1 + \psi_j y_2 = \alpha$, $\gamma_j y_1 + \tau_j y_2 = \beta$ for all $j \in s(\sigma_g^*(w))$.

(2) Compute the token T_{Pg} as

$$\begin{aligned} K_{u10} &= g_1 \prod_{k \in s(\sigma^*(w_g))} \left(h_k u_k^{\sigma^*(w_{gk})}\right)^{\lambda_k} \psi_k^{\gamma_k}, \\ K_{u20} &= g_2 \prod_{k \in s(\sigma^*(w_g))} \left(h_k u_k^{\sigma^*(w_{gk})}\right)^{\varphi_k} \psi_k^{\tau_k}, \\ K_{u30} &= g^{\alpha}, \\ K_{u40} &= g^{\beta}, \\ K_{u50} &= g^{-\sum_{k \in s(\sigma^*(w_g))}(v_k \alpha + t_k \beta)}. \end{aligned} \tag{11}$$

Let $T_{Pg} = (K_{u10}, K_{u20}, K_{u30}, K_{u40}, K_{u50})$. In the same way, BS can generate the T_{Ps} for partial query P_s. Then, the BS gets query tokens T_{Pg} and T_{Ps} and sends them to the requester S

$$\mathrm{BS} \longrightarrow S : T_{Pg}, T_{Ps}. \tag{12}$$

6.4. Data Query Phase. In this phase, S deposits its range query tokens to the DB to query their required data. After finish receiving the query tokens from S, the DB searches its database to find if there is a data ciphertext which matches S's query. For each data ciphertext, the DB checks that

$$f_{i0} = \frac{D_1 \cdot D_2 \cdot e(K_{u50}, C_{u50})}{e(K_{u10}, C_{u10}) \cdot e(K_{u20}, C_{u20})}, \tag{13}$$

where $D_1 = e(K_{u30}, \prod_{k \in s(\sigma_g^*(w))} C_{u30,k})$ and $D_2 = e(K_{u40}, \prod_{k \in s(\sigma_g^*(w))} C_{u40,k})$. If for all $k \in s(\sigma_g^*(w))$, $\sigma_g^*(w_k)$ and $\sigma_g(x_{ik})$ are equal, f_{i0} will be recovered. Similarly, f_{i1} will be recovered when $\sigma_s^*(w_k)$ and $\sigma_s(x_{ik})$ are equal at the corresponding position. Then, the BS can derive the coefficients of $f(x)$

by interpolation and hence calculate the secret $C_i = \Sigma_{v=1}^{t} f_{iv}L_v(0)$, where $L_v(0)$ is the Lagrange coefficient such as

$$L_v(x) = \frac{\prod_{j\neq v}(x - x_j)}{\prod_{j\neq v}(x_v - x_j)}. \quad (14)$$

Thus, the DB can recover C_i by using f_{i0} and f_{i1} and Lagrange coefficient. Then, DB distributes these data to S

$$\text{DB} \longrightarrow S : (C_i, \text{MAC}_i, \text{MAC}_i', N_i), (C_{i'}, \ldots, N_{i'}) \ldots . \quad (15)$$

6.5. Data Decryption and Verification. Note that, there are more than one query results which match and the range query tokens. When S received the query results from DB, it will send them to the BS for decryption and verification

$$S \longrightarrow \text{BS} : (C_i, \text{MAC}_i, \text{MAC}_i', N_i), (C_{i'}, \ldots, N_{i'}) \ldots . \quad (16)$$

The BS recovers original data from the ciphertexts by using its shared secret key k_i with S. The BS inputs data owner's identity N_i into its polynomial share $f(x, \text{BS})$ and obtains its shared key $k_i = f(N_i, \text{BS})$ with the N_i

$$\text{BS} : m_i = \text{Dec}_{k_i}(C_i), \ldots . \quad (17)$$

$\text{Dec}(\cdot)$ is the symmetric decryption algorithm corresponding to the opposite operation of $\text{Enc}(\cdot)$.

Then, the BS computes a new MAC as $\text{MAC}_1 = H\text{MAC}_{k_i}(m_i)$ by using the recovered m_i and k_i. The BS checks whether $\text{MAC}_1 = \text{MAC}_i$. If so, the data verification succeeds, which means that the recovered m_i is the original data; otherwise, the verification fails and m_i can be discarded. Next, the BS computes another MAC by using the received identity as $\text{MAC}_2 = H\text{MAC}_{k_i}(N_i)$. If $\text{MAC}_2 = \text{MAC}_i'$, the identity verification succeeds, which means that N_i is the original valid sender; otherwise, nothing will be returned. If all the verifications are successful, the BS distributes the correct query results to the requester in a secure channel as

$$\text{BS} \longrightarrow S : \{(m_i, N_i), \ldots\}. \quad (18)$$

As a result, the requester obtains the original data.

7. Security Analysis

In this section, we analyze the security properties of the proposed SSE scheme. In particular, following the security requirements discussed earlier, our analysis will focus on how the proposed SSE scheme can achieve the data confidentiality, data integrity, source authentication, and data query privacy.

(i) *The SN's data confidentiality is preserved in the proposed SSE scheme.* In the SSE, each SN's data m_i is encrypted by using the HVE-based encryption vectors $\sigma_g(\mathbf{x})$ and $\sigma_s(\mathbf{x})$. Anyone, including the cluster head CH and DB who does not know the BS's secret key and query tokens can not recover C_i and m_i from the ciphertexts CT_{i0} and CT_{i1}. Thus, the data confidentiality is preserved in the SSE.

(ii) *The SN's data integrity is preserved in the proposed SSE scheme.* In the SSE, each SN's data packet is followed by an MAC. The MAC is generated by using the shared secret key k_i between the SN and the BS. If any intermediate node, including the CH, forges or modifies the MAC, its malicious behavior can be detected during the data verification phase. The reason is that they have little knowledge of the secret keys k_i and m_i. Any modification will fail in the verification. Therefore, the data integrity is preserved in the SSE.

(iii) *The source authentication is preserved in the proposed SSE scheme.* In the SSE, except the data authentication message, MAC_i' is followed with the data packet, and another identity authentication message MAC_i' is also attached. After data verification, the BS also will check the source authentication by verifying if $\text{MAC}_i' = \text{MAC}_2 = H\text{MAC}_{k_i}(N_i)$ to authenticate sender's identity N_i. Similarly, any impersonation and modification will fail in the source authentication. By this means, the source authentication is preserved in the SSE.

(iv) *The data query privacy is achieved in the proposed SSE scheme.* In the SSE, when the query vectors in the query tokens match the HVE-base encryption vectors, the original data will not expose to the DB. The reason is that the secret keys needed for data decryption phase is kept secret. As described in the key generation phase, the polynomial share $h(i, y)$ for each node N_i is preloaded in N_i's memory. Only the BS and N_i can correctly generate the secret key $k_i = h(N_i, \text{BS}) = h(\text{BS}, N_i)$. Anyone who does not get the polynomial share $h(i, y)$ or $h(x, \text{BS})$ cannot compute the key k_i. As a result, even if the adversary $\mathscr{A}$ can get the data ciphertext and MAC message by eavesdropping, it also can not obtain any information about k_i and m_i. As a result, the data query privacy is preserved in the SSE.

Compared with the original OPE-based scheme [15], Buck-based scheme [6], and the original HVE-based scheme [20], as shown in Table 3, our SSE is the only one which can achieve all of the data confidentiality and intergrity, source authentication, and data query privacy. Furthermore, our SSE has no false positive results when querying the DB, which makes it more secure and efficient than the Buck-based scheme [6].

8. Conclusion

In this paper, we propose a secure searchable encryption scheme, named SSE, for urban sensing and querying to address the data security, source authentication, and range query problems of the sensed data in smart cities. The SSE allows the sensed data to be stored on a untrusted server in encrypted form. An authorized requester can obtain the correct ciphertexts when his authorized range query matches the HVE-based encryption predicate. With the help of the base

Table 3: Comparison of security properties.

Properties	SSE	OPE [15]	Buket [6]	HVE [20]
Confidentiality	Yes	partial	partial	Yes
Data integrity	Yes	No	No	No
Source authentication	Yes	No	No	No
Data query privacy	Yes	No	partial	partial
False positive	No	No	Yes	No

station, the data integrity and source authentication can be verified. Security analysis demonstrates that the SSE can achieve data confidentiality and integrity, source authentication, and data query privacy. For our future work, we intend to enhance our SSE to support fine-grained search with logical operations.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant nos. 61073189, 61272437, and 61202369, Innovation Program of Shanghai Municipal Education Commission nos. 13ZZ131 and 14YZ129, the Foundation Key Project of Shanghai Science and Technology Committee no. 12JC1404500, and the Project of Shanghai Science and Technology Committee no. 12510500700.

References

[1] A. Bartoli, J. Hernández-Serrano, M. Soriano, M. Dohler, A. Kountouris, and D. Barthel, "Security and privacy in your smart city," in *Proceedings of the Barcelona Smart Cities Congress*, December 2011.

[2] M. Chen, "Towards smart city: M2M communications with software agent intelligence," *Multimedia Tools and Applications*, vol. 67, no. 1, pp. 167–178, 2013.

[3] M. Wen, J. Li, Z. Yin, Y. Wang, and K. Chen, "A ntru based key generation and data transmission scheme for sensor networks," *Journal of Computational Information Systems*, vol. 8, no. 6, pp. 2417–2424, 2012.

[4] X. Wu, Y. Xu, C. Yuen, and L. Xiang, "A tag encoding scheme against pollution attack to linear network coding," *IEEE Transactions on Parallel and Distributed Systems*, 2013.

[5] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM '10)*, pp. 1–9, March 2010.

[6] B. Hore, S. Mehrotra, M. Canim, and M. Kantarcioglu, "Secure multidimensional range queries over outsourced data," *The VLDB Journal*, vol. 21, no. 3, pp. 333–358, 2012.

[7] P. P. Parikh, M. G. Kanabar, and T. S. Sidhu, "Opportunities and challenges of wireless communication technologies for smart grid applications," in *Proceedings of the IEEE Power and Energy Society General Meeting (PES '10)*, pp. 1–7, July 2010.

[8] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: security protocols for sensor networks," *Wireless Networks*, vol. 8, no. 5, pp. 521–534, 2002.

[9] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pp. 41–47, ACM, November 2002.

[10] J. Hwang and Y. Kim, "Revisiting random key pre-distribution schemes for wireless sensor networks," in *Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '04)*, pp. 43–52, ACM, October 2004.

[11] S. Zhu, S. Setia, and S. Jajodia, "LEAP: efficient security mechanisms for large-scale distributed sensor networks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*, pp. 62–72, ACM, October 2003.

[12] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing elliptic curve cryptography and RSA on 8-bit CPUs," in *Cryptographic Hardware and Embedded Systems—CHES 2004*, vol. 3156, pp. 119–132, Springer, Berlin, Germany, 2004.

[13] D. J. Malan, M. Welsh, and M. D. Smith, "A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography," in *Proceedings of the 1st Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON '04)*, pp. 71–80, IEEE, October 2004.

[14] Y. He and M. Li, "Cose: a query-centric framework of collaborative heterogeneous sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 9, pp. 1681–1693, 2012.

[15] A. Boldyreva, N. Chenette, and A. O'Neill, "Order-preserving encryption revisited: improved security analysis and alternative solutions," in *Advances in Cryptology—CRYPTO 2011*, vol. 6841, pp. 578–595, Springer, Berlin, Germany, 2011.

[16] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Theory of Cryptography*, vol. 4392, pp. 535–554, Springer, Berlin, Germany, 2007.

[17] E. Shi, J. Bethencourt, T.-H. H. Chan, D. Song, and A. Perrig, "Multi-dimensional range query over encrypted data," in *Proceedings of the IEEE Symposium on Security and Privacy (SP '07)*, pp. 350–364, May 2007.

[18] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order preserving encryption for numeric data," in *Proceedings of the ACM International Conference on Management of Data (SIGMOD '04)*, pp. 563–574, June 2004.

[19] S. de Capitani di Vimercati, S. Foresti, S. Paraboschi, G. Pelosi, and P. Samarati, "Efficient and private access to outsourced data," in *Proceedings of the 31st International Conference on Distributed Computing Systems (ICDCS '11)*, pp. 710–719, Springer, July 2011.

[20] J. H. Park, "Efficient hidden vector encryption for conjunctive queries on encrypted data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 10, pp. 1483–1497, 2011.

[21] M. Wen, K.-F. Chen, Y.-F. Zheng, and H. Li, "Reliable pairwise key-updating scheme for sensor networks," *Journal of Software*, vol. 18, no. 5, pp. 1232–1245, 2007.

[22] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Advances in Cryptology—CRYPTO 2001*, vol. 2139, pp. 213–229, Springer, Berlin, Germany, 2001.

[23] C. Blundo, A. de Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly secure key distribution for dynamic conferences," in *Advances in Cryptology—CRYPTO' 92*, vol. 740, pp. 471–486, Springer, Berlin, Germany, 1993.

[24] C. Zhang, R. Lu, X. Lin, P.-H. Ho, and X. Shen, "An efficient identity-based batch verification scheme for vehicular sensor networks," in *Proceedings of the 27th IEEE Communications Society Conference on Computer Communications (INFOCOM '08)*, pp. 246–250, IEEE, April 2008.