Secure and Verifiable Top-*k* Query in Two-Tiered Sensor Networks

Ting Zhou, Yaping Lin, Wei Zhang, Sheng Xiao, and Jinguo Li

Dept. of Information Science and Engineering, Hunan University,
Changshang 410082, China
{zhouting,yplin,zhangweidoc,xiaosheng,lijg1985}@hnu.edu.cn

Abstract. Two-tiered sensor networks have been widely adopted since they offer good scalability, efficient power usage and storage saving. Storage nodes, responsible for storing data from nearby sensors and answering queries from the sink, however, are attractive to attackers. A compromised storage node would leak sensitive data to attackers and return forged or incomplete query results to the sink. In this paper, we propose SVTQ, a Secure and Verifiable Top-*k* Query protocol that preserves both data confidentiality and integrity of query results. *To preserve data confidentiality*, we propose prime aggregation whereby storage nodes can process *top-k* queries precisely without knowing actual data values. *To preserve integrity of query results*, we further propose a novel scheme called differential chain that allows the sink to verify any forged or incomplete result. Both theoretical analysis and experimental results on the real-world data set confirm the effectiveness and efficiency of SVTQ protocol.

Keywords: Two-tiered sensor networks, Data confidentiality, Prime aggregation, Integrity, Differential chain.

1 Introduction

1.1 Motivation

Two-tiered sensor networks have been widely adopted since they offer good scalability, efficient power usage and storage saving. In this paper, we focus on a two-tiered sensor network as illustrated in Fig. 1, where resource-rich storage nodes act as an intermediate tier between sensor nodes and the sink. The storage nodes store data from their nearby sensors and process queries from the sink.

Compared with traditional sensor networks, two-tiered sensor networks have three major advantages. First, sensors periodically submit their collected data to their nearby storage node in one hop instead of sending them to the sink via multiple hops, which saves power for energy-limited sensors. Second, storage nodes store sensor collected data for future retrieval and data analysis, which saves storage space for memory-constrained sensors. Third, when issuing a query, instead of flushing the whole sensor network, the sink only needs to communicate with storage nodes, and storage nodes only need to process queries over the data stored on them locally, which makes the query more efficient. Two-tiered sensor networks were first introduced by

T. Zia et al. (Eds.): SecureComm 2013, LNICST 127, pp. 19–34, 2013.

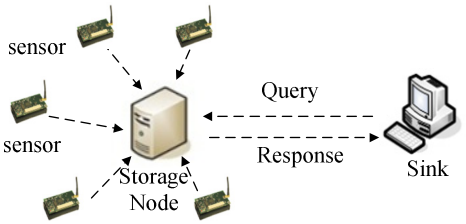


Fig. 1. Architecture of two-tiered sensor networks

Sylvia Ratnasamy [13], and then have been widely adopted for various applications [5], [15], [17], [20]. Several commercial products of storage nodes, such as RISE [14] and StarGate [19], have been available and widely used.

However, the central role of storage nodes in this tiered framework makes them attractive to attackers when the network is deployed in a hostile environment. A compromised storage node poses great threats to the network. First, a compromised storage node would leak sensitive information collected from sensors to attackers. Second, a compromised storage node would also return forged or incomplete query results to the sink. It is especially dangerous when the query results are used to make critical decision such as military actions. Therefore, a secure and verifiable query protocol for two-tiered sensor networks is imperative.

As a typical query type in two-tiered sensor networks, *top-k* query asks for data items whose numerical attributes are among the k highest of all data items [21], which is important for monitoring extreme conditions. Therefore, this paper aims to design a secure and verifiable *top-k* query protocol for two-tiered sensor networks. For data confidentiality, the proposed query protocol should enable storage nodes to process queries correctly without knowing actual data values, so that compromising a storage node will not lead to the leakage of sensitive data. For integrity of query results, the query protocol should allow the sink to verify whether the storage nodes has injected forged data into or omitted some qualified data items from query results.

1.2 Technical Challenges

There are two key challenges in designing a secure and verifiable *top-k* query protocol for two-tiered sensor networks. First, to prevent a compromised storage node from disclosing sensitive data to attackers, each sensed data should be encrypted before being sent to storage nodes, hence, the storage nodes need to process queries over encrypted data items without knowing their actual data values. Second, upon receiving a query result from a storage node, the sink needs to verify whether the query results indeed contain the *top-k* data items and do not contain any forged data.

1.3 Limitations of Prior Arts

Although important, only in recent years has secure *top-k* query in two-tiered sensor networks become a focus of research. Zhang & Shi firstly propose schemes for

verifiable *top-k* query in their recently seminal work [21]. By exploiting crosscheck approach, the integrity of query results can be well preserved. Nevertheless, data confidentiality is not taken into consideration. A subsequent solution to secure *top-k* query was presented by Liao and Li [11], which covers both data confidentiality and integrity of query results. All the schemes proposed in [11] are based on the system model where each sensor submits only one sensed data to storage node each time. However, it is a more general case in real-world applications where each sensor has multiple sensed data per submission, as adopted in prior arts [3], [16], [18], [21]. Liao's scheme for data confidentiality preservation would be unworkable when performed on this general system model. In addition, Liao's scheme does not enable storage nodes to obtain precise query results while ensuring data confidentiality. Up to now, no research effort was conducted on both confidentiality and integrity preserving *top-k* query in the general system model of two-tiered sensor networks.

1.4 Our Approach and Major Contributions

In this paper, we propose SVTQ, a secure and verifiable *top-k* query protocol for two-tiered sensor networks. To preserve data confidentiality, we propose a novel scheme called prime aggregation whereby storage nodes can process queries correctly without knowing actual data values. To preserve integrity of query results, we propose a differential chain, a novel scheme which enables the sink to verify the authenticity and completeness of query results. The major contributions of this paper are listed as follows:

(1) To the best of our knowledge, this paper is the first that considers both data confidentiality and integrity issues when processing *top-k* queries in the general system model of two-tiered sensor networks.

(2) We propose a novel data confidentiality preserving scheme which can precisely obtain *top-k* query results without disclosing any sensitive information to storage nodes.

(3) We introduce a data storage scheme which allows the sink to verify any forged or incomplete query result.

(4) We evaluate our solutions on a real-world data set, and the results show that SVTQ achieves confidentiality and integrity goals efficiently.

The rest of this paper is organized as follows. In Section 2, we give a brief review of the related work. Section 3 describes the system model and the threat model. In section 4 and 5, we give a detailed description of our data confidentiality and integrity preserving scheme respectively. In Section 6, we discuss the security and performance of our proposed schemes. We present our performance evaluation and experimental results in section 7 and conclude this paper in section 8.

2 Related Work

2.1 Secure Range Query in Two-Tiered Sensor Networks

Secure range query in two-tiered sensor networks has attracted much attention in recent years [3], [16], [18]. To preserve data confidentiality, Sheng & Li [16] and Shi *et al.* [18] adopt the bucket partitioning scheme first introduced by Hacigumus *et al.* [8]. The basic idea of bucket partitioning is to divide the domain of data value into multiple buckets, after collecting data items from environment, each sensor firstly distributes the collected data into a corresponding bucket, encrypts data items in each bucket, and then sends each encrypted data along with its bucket ID to the closest storage node. When the sink wants to execute a range query, it first converts the query into a smallest set of bucket IDs and then sends the set to the storage node. Once receiving the queried set of bucket IDs, the storage node first finds all the encrypted data items that fall into these buckets, and then reports them to the sink as the query result. However, as pointed out in [9], the bucket partitioning scheme allows a compromised storage node to make a reasonable estimation on both sensed data and queries. To address this problem, Chen & Liu proposed SafeQ [3], a prefix based scheme to encode both data and queries such that the aforementioned estimation can be avoided. In SafeQ, after collecting n data items from the environment, each sensor firstly converts the n data items into $n + 1$ ranges, and then employs prefixes to represent these ranges before sending them to storage node. However, suffering from the inherent drawback of prefix membership verification technique, SafeQ usually needs a series of prefixes to represent a range, which is unfavourable for resource-limited sensors.

To preserve integrity of query results, Sheng & Li [16] proposed an encoding technique where each sensor generates a distinct encoding number for the bucket that has no data item, these encoding numbers are used by the sink to verify the integrity of query results. However, this technique will introduce extra communication overheads by sending the encoding number for these empty buckets. To address this problem, Shi *et al.* [18] proposed a spatiotemporal crosscheck approach. In their scheme, each sensor uses a bit map to represent which buckets have data, and then broadcasts its bit map to nearby sensors. Each sensor attaches the bit maps received from others to its own data and then encrypts them together. The sink verifies the integrity of query result from a sensor by examining the bit maps from its nearby sensors. However, this scheme would be efficient only in the event detection scenarios since data broadcast would introduce considerable communication overheads. With respect to this problem, Chen & Liu [3] proposed a technique called neighborhood chain, by concatenating the neighboring data items, the sink can detect the integrity of query results efficiently.

2.2 Secure Top-*k* Query in Two-Tiered Sensor Networks

Secure *top-k* query in two-tiered sensor networks has been recently studied [11], [21]. Zhang & Shi firstly proposed three schemes for verifiable *top-k* query in their recent

work [21]. All their schemes share the same basic idea that each data item is attached with a Message Authentic Code (MAC) such that injecting forged data can be easily detected. Specially, in their later two schemes, sensors exchange data information by broadcasting their highest data score to others. Each sensor embeds the received information into its own data and then sends these new data items along with MACs to its closest storage node. The sink then verifies the authenticity and completeness of query results by examining the MACs and the information extracted from each data. By using this crosscheck approach, the integrity of query results can be well preserved. However, as we aforementioned, the broadcast mechanism would introduce considerable communication overheads for energy-constrained sensors. Furthermore, data confidentiality is not taken into consideration in [21].

A subsequent solution to secure *top-k* query in two-tiered sensor network was proposed by Liao and Li [11], which covers both data confidentiality and integrity of query results. By using the revised order-preserving symmetric encryption (*OPSE*) proposed by Boldyreva *et al.* [1], the storage nodes can process *top-k* query as efficient as for the unencrypted data items. However, the simply use of *OPSE* would incur order-relation and distance-relation privacy leakage. To overcome this problem, Liao *et al.* further propose a secret perturbation scheme. The basic idea of secret perturbation is to randomly select some sensors, and then perturb the data of these sensors by adding a secret data to the original one before encrypting them with *OPSE*. However, this scheme would lead to imprecise query results, which means the query results would contain data items that do not satisfy the query. What's more, all the schemes proposed in [11] are based on the system model where each sensor submits one sensed data to storage node each time. This means that the *top-k* queries would be performed on the data set where each sensor only has one data. However, it is a more general case in real-world applications where each sensor has multiple data items per submission, as assumed in [3], [16], [18], [21]. The data confidentiality preserving scheme proposed in [11] would be unworkable when performed on this general system model. In contrast, this paper aims at designing a *top-k* query protocol in the general system model as adopted in prior arts [3], [16], [18], [21], which preserves both data confidentiality and integrity of query results.

3 Models and Problem Statement

3.1 System Model

We assume a similar system model as in [3], [16], [18], [21]. The architecture of this two-tiered sensor network is shown in Fig. 1. Specifically, each storage node is in charge of a cell composed of many sensors, storage nodes are often resource-rich in the aspects of energy, storage and computation while sensors are usually resource-constrained in every regard. Sensors collect data from the environment and periodically submit their collected data to storage nodes. Storage nodes store data received from their nearby sensors and process queries from the sink. Without loss of generality, we assume that all sensors and storage nodes are loosely synchronized. We divide time into fixed time intervals, and every n intervals form an epoch. Each sensor sends its data to the storage node at the end of each epoch as follows.

$$S_i \rightarrow \text{Storage Node}: \ i, t, \{d_{i,1}, \ldots, d_{i,n}\}$$

where i is the sensor ID and t is the sequence number of the epoch during which n data items $\{d_{i,1}, \ldots, d_{i,n}\}$ are collected. Similarly, we consider the following query mode.

$$\text{Sink} \rightarrow \text{Storage Node}: \quad Q = (A, t, k)$$

where A denotes the ID set of queried sensors, t is the queried time epoch and k denotes the number of data items the sink asks the storage node to return. To sum up, $Q = (A, t, k)$ denotes that the sink asks for the k highest data items generated in a queried set A during epoch t.

3.2 Threat Model and Security Goals

Similar with prior arts, we assume that storage nodes are vulnerable to be compromised while sensors and the sink are always trustworthy. Due to the important role of storage nodes in two-tiered sensor networks, compromising a storage node will lead to great damage to the system. First, once a storage node is compromised, the large quantity of confidential data stored on the storage node will be leaked to the attackers. Second, after receiving a query from the sink, the compromised storage node can also be manipulated to return forged or incomplete query results to the sink. Therefore, this paper aims to achieve the following security goals.

Data Confidentiality preservation: To enable storage nodes to process queries correctly over encrypted data without knowing actual data values, so that compromising a storage node will not lead to the leakage of any sensitive data.

Integrity preservation: To enable the sink to verify the authenticity and completeness of query results. The authenticity check is to detect forged data items in query results while the completeness check is to make sure that no qualifying data items are maliciously omitted by compromised storage nodes. Thus, any misbehavior of a compromised storage node can be detected by the sink.

4 Confidentiality Preservation for Sensed Data

In order to preserve data confidentiality, it seems that the simplest way is to encrypt data before sending them to the storage node. But a subsequent challenge is how to process a query over encrypted data without revealing plaintext data.

4.1 Prefix Membership Verification

Prefix membership verification was first introduced in [4] and later formalized in [12]. The basic idea of this technique is to convert the question of whether a number is in a range to another question of whether two sets share a prefix. Given a number x whose binary format is $b_1b_2\ldots b_w$, where w is the bit length of number x. The prefix family of number x is defined as $F(x) = \{b_1b_2\ldots b_w, b_1b_2\ldots b_{w-1}*, \ldots, b_1* \ldots *, * \ldots *\}$.

For example, the prefix family of number 11 is $F(11)$ = {1011, 101*, 10**, 1***, ****}. Similarly, a range can be converted into a minimum set of prefixes which we call the range prefix of this range. We use $R[d_1, d_2]$ to denote the range prefix of range $[d_1, d_2]$. For example, $R[10, 15]$ = {101*, 11**}. Given a number x and a range $[d_1, d_2]$, if $F(x) \cap R[d_1, d_2] \neq \emptyset$, we can draw the conclusion that $x \in [d_1, d_2]$ according to the theory proposed in [12].

Inspired by works [4] and [12], we propose to make use of the prefix membership verification technique to meet the data confidentiality preserving goal. However, there are still two challenges need to be overcome before we can apply this technique to our problem. First, the aforementioned method is just used to determine whether a number belongs to a range, how to make comparison between two encrypted data is still a difficult question. Second, given a number d_1 and a range $[d_2, d_3]$, where d_1, d_2, and d_3 are numbers with w bits. We need $w + 1$ prefixes to denote the prefix family of d_1 and $2w$-2 prefixes to denote the range prefix of this range in the worst case [7]. This is heavy-laden for resource-constrained sensors if so many prefixes need to be submitted for each sensing data for the sake of data comparison. Thus, how to reduce these massive overheads remains another question.

4.2 Prime Aggregation

To meet aforementioned challenges, we propose prime aggregation, a novel scheme which enables the comparison between two encrypted data items while fewer additional overheads are introduced. The basic idea of prime aggregation is to aggregate multiple prefixes in a prefix set into a single number with the aid of primes, which can be done in two steps. First, map each prefix in prefix set to a unique prime. Second, perform multiplication on the primes obtained from each prefix set.

For simplicity, the *Prime* aggregation result from prefix *Family* will be denoted by *PF*, while the *Prime* aggregation result from *Range* prefix will be denoted by *PR*. Then we have the following theorem.

Theorem 4.1: *Given a number x and a range* $[y, z]$, $x \in [y, z]$ *if and only if the following inequality holds:*

$$gcd\,(PF(x), PR([y, z])) \neq 1\ . \tag{1}$$

Proof: Let f_i and r_j be the prefix in the prefix family and range prefix respectively, while the total number of prefixes in the prefix family and range prefix are denoted by V and M respectively. Then we have

$$F(x) = \bigcup_{i=1}^{V} f_i\ ,\ \ R([y, z]) = \bigcup_{j=1}^{M} r_j\ .$$

Similarly, let pf_i and pr_j be the corresponding prime of f_i and r_j respectively, then

$$PF(x) = \prod_{i=1}^{V} pf_i\ ,\ \ PR([y, z]) = \prod_{j=1}^{M} pr_j\ .$$

If $x \in [y, z]$, we have

$$F(x) \cap R([y, z]) = f_h = r_l \neq \emptyset,\ \ h \in [1, V]\ ,\ \ l \in [1, M]\ .$$

Then $pf_h = pr_l$

i.e. $x \in [y, z] \Rightarrow gcd(PF(x), PR([y, z])) = pf_h = pr_l \neq 1$.

The inversion of this expression can be proved similarly, thus

$$x \in [y, z] \Leftrightarrow gcd(PF(x), PR([y, z])) \neq 1.$$

Next, we introduce the detailed process for applying prime aggregation to our scheme.

Before distributing all sensors to their working sites, a sequence of prime numbers will be pre-generated and stored in all sensors. For convenience of mapping, each prefix in prefix set would be firstly numericalized before mapped to a corresponding prime.

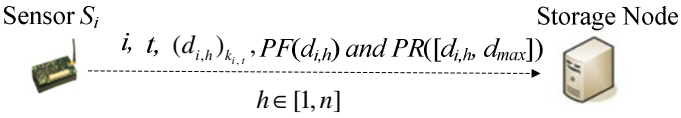


Fig. 2. Data submission for a sensor node

The basic idea of prefix numericalization is to convert a prefix into a corresponding number. Given a prefix $b_1 b_2 \cdots b_h$ * ⋯ *of w bits, we first insert 1 between b_h and the symbol * to separate $b_1 b_2 \cdots b_h$ and * ⋯ *, and then replace each * with 0. Note that if there is no symbol * in the prefix, we will add 1 at the end of this prefix. For example, prefix {1***} will be converted into {11000} = 24 while {1110} will be converted into {11101} = 29 after numeralization. The detailed process of prefix numeralization falls out of scope of this paper. A formal definition of this process can be found in [2].

For mapping each prefix to a unique prime, we introduce a pseudorandom function in our scheme. The pseudorandom number generated by the pseudorandom function acts as a medium for mapping each prefix to a unique prime. Specifically, at the beginning of each epoch, a list of pseudorandom numbers will be generated using a pseudorandom function shared by all sensors. Note that the seed for this pseudorandom function varies with epoch changing, *i.e.*, $seed_t = hash\ (seed_{t-1})$. When mapping each prefix to a prime, each sensor firstly finds the corresponding pseudorandom number according to the value of each numericalized prefix, *e.g.*, if the value of a numericalized prefix is 12, the sensor finds the 12*th* pseudorandom number in the pseudorandom number list. Then the value of this pseudorandom number will be used as the address for indexing the prime stored on each sensor.

For the second step of prime aggregation, we perform multiplication to the obtained primes. Note that each large number is regarded as a string in our scheme, and we adopt the large number multiplication algorithms proposed in [6] to perform multiplication. Thus, the product of primes can be any length and the overflow can be avoided.

4.3 Data Submission

Let $d_{i,1}, \ldots, d_{i,n}$ be the data items collected by sensor S_i during epoch t, where each data item belongs to range (d_{min}, d_{max}). Here d_{min} and d_{max} , known to both sensors and the sink, denote the public lower and upper bounds of sensor collected data. Fig. 2 illustrates the information sent to storage node by sensor S_i at the end of epoch t.

Upon collecting n data items, sensor node S_i performs the following steps:

(1) Sort the n data items in descending order, *i.e.*, $d_{i,1} > d_{i,2} > \ldots > d_{i,n}$. For simplicity, we assume that all data items are different from each other.

(2) Compute the prefix family $F(d_{i,h})$ and the range prefix $R([d_{i,h}, d_{max}])$ for each data item, where $h \in [1, n]$, and then numericalize them.

(3) Perform prime aggregation to the numericalized prefixes. We use $PF_{i,h}$ to represent the aggregation result of $d_{i,h}$ while $PR_{i,h}$ to denote that of range $[d_{i,h}, d_{max}]$.

(4) Encrypt each data with the secret key $k_{i,t}$, shared between each sensor and the sink, *i.e.*, compute $(d_{i,1})_{k_{i,t}}$, …, $(d_{i,n})_{k_{i,t}}$, where $k_{i,t}$ is generated using an embedded hash function, *i.e.*, $k_{i,t} = hash\ (k_{i,t-1})$.

(5) Send the encrypted data items along with the prime aggregation results to storage node, *i.e.*, send $\{[(d_{i,1})_{k_{i,t}}, PF_{i,1}, PR_{i,1}], \cdots, [(d_{i,n})_{k_{i,t}}, PF_{i,n}, PR_{i,n}]\}$ to its closest storage node.

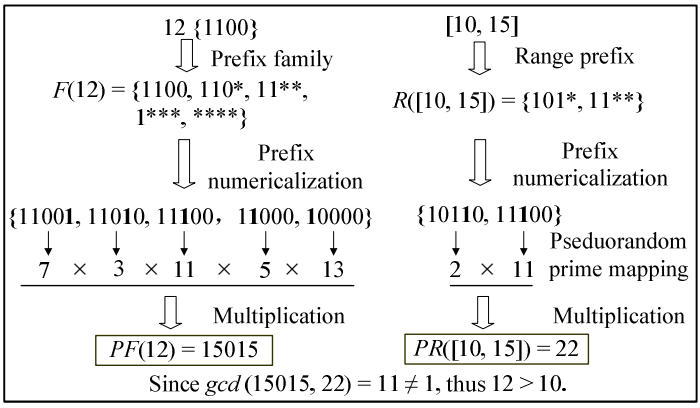


Fig. 3. An example of data comparison using prime aggregation

4.4 Query Processing

After receiving a query $Q = (A, t, k)$ from the sink, the storage node finds the largest k data items of queried set A based on theorem 4.1.

Given two encrypted data items $(d_{i,h})_{k_{i,t}}$ of sensor S_i and $(d_{j,l})_{k_{j,t}}$ of sensor S_j, where $i, j \in A$ and $h, l \in [1, n]$. Storage nodes compare these two encrypted data items by checking whether the following expression holds.

$$gcd\ (PF_{i,h}, PR_{j,l}) \neq 1$$

If the above expression holds, then

$$d_{i,h} \in [d_{j,l}, d_{max}]$$

Namely

$$d_{i,h} > d_{j,l}.$$

By doing this, storage nodes can make comparison between two numbers without knowing their actual values. Thus the *top-k* results can be precisely obtained.

As an illustrative example, shown in Fig. 3, when given two encrypted data $(12)_{k_{i,t}}$ and $(10)_{k_{j,t}}$, storage nodes compare these two data items by verifying whether *gcd* (*PF*(12), *PR*([10, 15])) ≠ 1. Here, we assume the public upper bound of data is 15.

Algorithm 1. *diff_C* ($d[]$, d_{max}, d_{min})

```
1.  sort (d) by descending order; n = d.size
2.  if   n = 1 then
3.       D[1] ← (d_max – d[1]) ‖ d[1] ‖ (d[1] – d_min); return(D)
4.  else if  n = 2 then
5.       D[1] ← (d_max – d[1]) ‖ d[1] ‖ (d[1] – d[2])
6.       D[2] ← d[2] ‖ (d[2] – d_min); return(D)
7.  else
8.       for  i = 2: n-1
9.          D[i] ← d[i] ‖ (d[i]-d[i + 1])
10.      end for
11.      D[1] ← (d_max – d[1]) ‖ d[1] ‖ (d[1] – d[2])
12.      D[n] ← d[n] ‖ (d[n] – d_min); return (D)
13. end if
```

5 Integrity Preservation for Query Results

In this section, we propose differential chain, a novel data storage scheme that enables the sink to verify any forged or incomplete query result.

5.1 Differential Chain

The basic idea of differential chain is to embed the difference of two adjacent data items into the prior one, hence, data items are linked with each other just like a chain. The procedure for transforming the list of sensed data to a differential chain is shown in Algorithm 1. Here, " ‖ " denotes the concatenation of data items.

After collecting n data items at the end of each epoch, sensor S_i firstly converts these sensed data into a corresponding differential chain, and then sends the encrypted differential chain of $(D_{i,1})_{k_{i,t}}$, $(D_{i,2})_{k_{i,t}}$, …, $(D_{i,n})_{k_{i,t}}$, instead of $(d_{i,1})_{k_{i,t}}$, $(d_{i,2})_{k_{i,t}}$, …, $(d_{i,n})_{k_{i,t}}$, to its closest storage node.

For simplicity, we assume that all data items are different from each other. In fact, when come to the case in which some sensed data are the same, we can adjust our scheme by further embedding the sequence number of the sensed data into its predecessor. Hence, the embedded information for each data item will be unique.

5.2 Query Response

Note that if sensor S_i has β_i data items satisfying a *top-k* query, they must be the first β_i data items since data items are ordered before being sent to storage node. Then:

If $\beta_i = 0$, the storage node need to respond a *VI* (*Verification Information*) for sensor S_i for the final verification.

Storage → Sink : $VI_i = \{ i, (D_{i,1})_{k_{i,t}} \}$

If $\beta_i \neq 0$, we call sensor S_i a qualified sensor, thus the satisfied *QR* (*Query Result*) of sensor S_i would be:

Storage → Sink : $QR_i = \{ i, (D_{i,1})_{k_{i,t}}, \ldots, (D_{i,\beta_i})_{k_{i,t}} \}$.

After receiving a query result from storage node, the sink first decrypts QR_i for each qualified sensor using the secret key shared between each sensor and the sink, then obtains the final *top-k* result by extracting the embedded information from each data item. Finally, the sink verifies the authenticity and completeness of the query result by checking whether the differential chain of each sensor is complete or not. Only when the query result has passed the verification, will the result be received.

6 Analysis

6.1 Data Confidentiality Analysis

If a storage node is compromised, it wouldn't disclose any sensed data to attackers since each data is encrypted using a secret key only known by sensor itself and the sink. The only choice left for an attacker to obtain data information is the two aggregation results attached to each data item. However, by mapping each prefix to a unique prime randomly, attackers can only obtain the exact mapping relationship between prefixes and primes with probability $\frac{1}{(2^{w+1})!}$, where w is the bit length of each sensed data. Furthermore, the seed for the pseudorandom function varies with epoch changing, this means the mapping relationship between prefixes and primes will also change with epoch changing, which makes the inference even more difficult. Therefore, even if storage nodes are compromised, confidentiality of the collected data would be preserved.

6.2 Integrity Analysis

Theorem 6.1: *Our scheme can enable the sink to detect any forged or incomplete top-k query result.*

Proof: Consider a sensor S_i who has β_i data items satisfying the query, *i.e.*, $QR_i = \{(i)_{k_t}, (D_{i,1})_{k_{i,t}}, \ldots, (D_{i,\beta_i})_{k_{i,t}}\}$. Since data items in QR_i are constructed into a chain,

inserting forged data into or omitting qualified data from QR_i can be easily detected. An alternative way for a compromised storage node is to replace some qualified data of one sensor, say D_{i,β_i} of S_i, with some unqualified data of another sensor, say D_{j,β_j+1} of S_j, *i.e.*, $QR_i = \{ (i)_{k_t}, (D_{i,1})_{k_{i,t}}, \ldots, (D_{i,\beta_i-1})_{k_{i,t}} \}$ and $QR_j = \{ (j)_{k_t}, (D_{j,1})_{k_{j,t}}, \ldots, (D_{j,\beta_j})_{k_{j,t}}, (D_{j,\beta_j+1})_{k_{j,t}} \}$. However, after extracting the embedded information from each data, the sink will know the existence of d_{i,β_i} with the difference extracted fromD_{i,β_i-1}. Since there is still a data larger than one of the data in query result, *i.e.*, $d_{i,\beta_i} > d_{j,\beta_j+1}$, the query result will be deemed as untrustworthy and discarded. Thus, any forged or incomplete query result can be detected.

6.3 Performance Analysis

In this paper, we use the following performance metrics to analyze and evaluate our proposed schemes.

C_{tra}-*Transmission Consumption*: the extra communication costs during a data submission for each sensor. The cost for transmitting actual data items, sensor ID and the epoch number will not be considered since it is inevitable for any scheme.

C_{spa}-*Space Consumption*: the space costs for a storage node to store the extra information received from the whole cell within an epoch.

C_q-*Query Consumption*: the extra communication costs for a storage node to respond a query. The cost for sending the satisfied *k* data items and the qualified node IDs will not be considered similarly.

We assume that each cell contains *N* sensors and each sensor S_i collects *n* data items during an epoch as we aforementioned, where $i \in [1, N]$. Recall that in our scheme, each data item is accompanied with two prime aggregation results when being submitted to storage node. These prime aggregation results contribute to the extra transmission consumption of our scheme. Specifically, let $lf_{i,h}$ and $lr_{i,h}$ be the bit length of the aggregation results of $PF_{i,h}$ and $PR_{i,h}$ respectively, then we can derive the extra transmission consumption C_{tra} for sensor S_i.

$$C_{tra} = \sum_{h=1}^{n} \left(lf_{i,h} + lr_{i,h} \right). \tag{1}$$

Similarly, we can derive the extra space consumption C_{spa} for a storage node.

$$C_{spa} = \sum_{i=1}^{N} \sum_{h=1}^{n} \left(lf_{i,h} + lr_{i,h} \right). \tag{2}$$

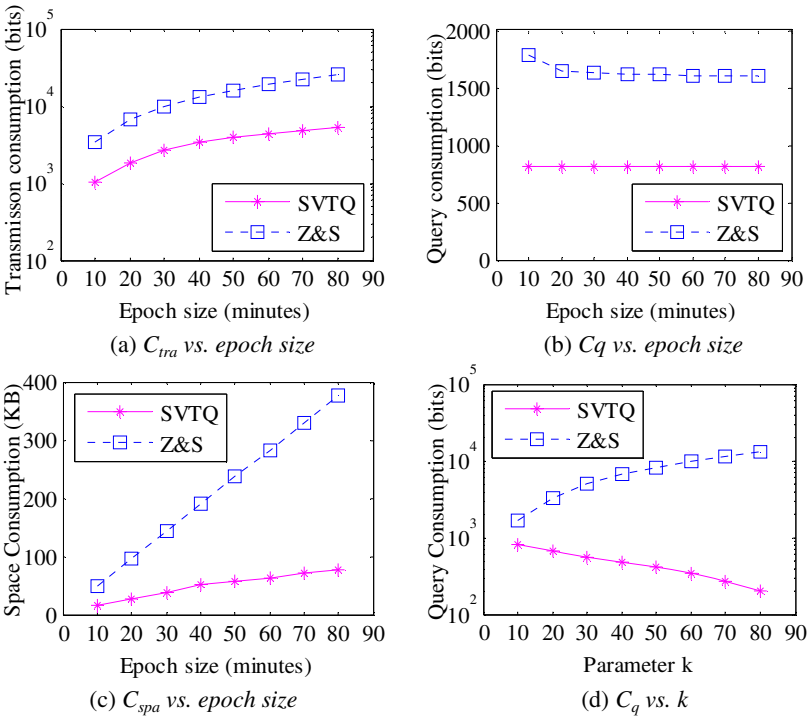
Note that in our scheme, except for the satisfied *top-k* query result, the storage node is asked to return the largest data item of each unqualified sensors for the sake of final integrity verification, which contributes to the additional query consumption of our scheme. Let l_{id} and l_d be the bit length of the encrypted node ID and the encrypted data respectively, and δu be the number of unqualified sensors, then we have

$$C_q = \delta_u \left(l_{id} + l_d \right). \tag{3}$$

7 Performance Evaluation

7.1 Evaluation Methodology

To compare SVTQ with the state-of-the-art presented by Zhang & Shi [21] which we call Z&S scheme, we implemented both schemes and performed side-by-side comparison on a large real data set from Intel Lab [10], which is collected from 54 sensors during one month. For easy division, we selected data from 45 sensors in our experiments and evenly divided these 45 sensors into 3 cells. Specially, for Z&S scheme, each cell is further divided into 3 subcells.



(a) C_{tra} *vs. epoch size* (b) *Cq vs. epoch size*

(c) C_{spa} *vs. epoch size* (d) C_q *vs. k*

Fig. 4. Additional communication and space consumption

7.2 Evaluation Setup

We adopted DES cipher in SVTQ as the encryption algorithm to encrypt sensor collected data. Since each block size in DES is 64 bits, there is enough space to embed the difference between two adjacent sensed data before we encrypted each data item. For the implementation of Z&S scheme, we adopted MD5 with 16-bit keys for massage authentication code (MAC) as mentioned in their scheme. We experimented on different size of epochs ranging from 10 minutes to 80 minutes. We also generated 8 different queries ranging from *top-10* to *top-80* to verify the impact of parameter *k*

on the communication cost of both schemes. We performed 1,000 times for each *top-k* query and took the average value as our final experimental results.

7.3 Result Analysis and Summary

Through our side-by-side comparison, we can see that SVTQ outperforms Z&S scheme in terms of power and space consumption while preserving the data confidentiality.

Fig. 4. (a) shows that as epoch size increases, both the transmission consumption of SVTQ and Z&S scheme grow up. This is clear since larger epoch means more data items are collected within an epoch, hence more additional information is needed to be submitted.

Fig. 4. (b) illustrates the additional query costs for a storage node to respond a query. As we can see from the figure, the C_q of our scheme remains unchanged with the variation of epoch size. This is of no surprise since the additional query consumption of our scheme is independent of epoch size and mainly caused by the unqualified sensors. While for Z&S scheme, the C_q is inversely proportional to epoch size. This is because the larger epoch size, the fewer IDs are attached to each data item as mentioned in [21].

Fig. 4. (c) demonstrates the extra space consumption on a storage node during an epoch. We can learn from the figure that the space consumption in both schemes grows up with the increase of epoch size. The reason is obvious since larger epoch size means more data items are collected by sensors within an epoch, and therefore more additional information is needed to be stored.

Fig. 4. (d) shows the impact of parameter k on the additional costs for a storage node to respond a query. We can see that the C_q of SVTQ decreases with the growth of k while Z&S scheme is on the contrary. This is because larger k implies more qualified sensors but fewer unqualified ones, and therefore less additional information is needed to be responded in SVTQ. While for Z&S scheme, larger k means more additional MACs and redundant IDs returned with each qualified data item, hence, the extra query costs of Z&S scheme grow up when parameter k increases.

8 Conclusions

In this paper, we propose SVTQ, a novel and efficient query protocol for processing *top-k* query in two-tiered sensor networks. SVTQ can precisely process *top-k* queries while preserving data confidentiality. SVTQ also enables the sink to detect any forged or incomplete query result efficiently. Experiments on the real-world data set show that SVTQ significantly outperforms the state-of-the-art in terms of both communication and storage consumption while preserving data confidentiality.

Acknowledgements. This work is supported in part by the National Natural Science Foundation of China (Project No. 61173038) and Important National Science & Technology Support Projects of China (Project No. 2012BAH09B02).

References

1. Boldyreva, A., Chenette, N., O'Neill, A.: Order-Preserving Encryption Revisited: Improved Security Analysis and Alternative Solutions. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 578–595. Springer, Heidelberg (2011)
2. Chang, Y.K.: Fast binary and multiway prefix searches for packet forwarding. Computer Networks 51(3), 588–605 (2007)
3. Chen, F., Liu, A.X.: SafeQ: Secure and Efficient Query Processing in Sensor Networks. In: Proceedings of the 29th IEEE International Conference on Computer Communications, INFOCOM 2010, pp. 1–9. IEEE, California (2010)
4. Cheng, J., Yang, H., Wong, S.H.Y., Zerfos, P., Lu, S.: Design and Implementation of Cross-Domain Cooperative Firewall. In: Proceedings of the IEEE International Conference on Network Protocols, ICNP 2007, pp. 284–293. IEEE, Beijing (2007)
5. Desnoyers, P., Ganesan, D., Li, H., Shenoy, P.: Presto: A predictive storage architecture for sensor networks. In: 10th Workshop on Hot Topics in Operating Systems, HotOS 2005, pp. 23–28. USENIX, New Mexico (2005)
6. Fürer, M.: Faster integer multiplication. In: Proceedings of the 39th ACM Symposium on Theory of Computing, STOC 2007, pp. 57–66. ACM, New York (2007)
7. Gupta, P., McKeown, N.: Algorithms for packet classification. IEEE Network 15(2), 24–32 (2001)
8. Hacigümüs, H., Iyer, B., Li, C., Mehrotra, S.: Executing SQL over encrypted data in the database-service-provider model. In: Proceedings of the ACM Conference on Management of Data, SIGMOD 2002, pp. 216–227. ACM, New York (2002)
9. Hore, B., Mehrotra, S., Tsudik, G.: A privacy-preserving index for range queries. In: Proceedings of the 30th International Conference on Very Large Data Bases, VLDB 2004, pp. 720–731. ACM, Toronto (2004)
10. Intel lab data, `http://db.csail.mit.edu/labdata/labdata.html`
11. Liao, X., Li, J.: Privacy-preserving and secure top-k query in two-tier wireless sensor network. In: Proceedings of the IEEE Global Communications Conference, GLOBECOM 2012, pp. 335–341. IEEE, California (2012)
12. Liu, A.X., Chen, F.: Collaborative enforcement of firewall policies in virtual private networks. In: Proceedings of the 27th ACM Symposium on Principles of Distributed Computing, PODC 2008, pp. 95–104. ACM, New York (2008)
13. Ratnasamy, S., Karp, B., Shenker, S., Estrin, D., Govindan, R., Yin, L., Yu, F.: Data-Centric Storage in Sensornets with GHT, a Geographic Hash Table. Mobile Networks and Applications 8(4), 427–442 (2003)
14. Rise project, `http://www.cs.ucr.edu/rise`
15. Sheng, B., Li, Q., Mao, W.: Data storage placement in sensor networks. In: Proceedings of the 7th ACM International Symposium on Mobile Ad hoc Networking and Computing, MobiHoc 2006, pp. 344–355. ACM, New York (2006)
16. Sheng, B., Li, Q.: Verifiable Privacy-Preserving Range Query in Two-Tiered Sensor Networks. In: Proceedings of the 27th IEEE International Conference on Computer Communications, INFOCOM 2008, pp. 46–50. IEEE, Phoenix (2008)
17. Sheng, B., Tan, C.C., Li, Q., Mao, W.: An Approximation Algorithm for Data Storage Placement in Sensor Networks. In: Proceedings of the International Conference on Wireless Algorithms, Systems and Applications, pp. 71–78. IEEE, Chicago (2007)

18. Shi, J., Zhang, Y., Zhang, Y.: Secure Range Queries in Tiered Sensor Networks. In: Proceedings of the 28th IEEE International Conference on Computer Communications, INFOCOM 2009, pp. 945–953. IEEE, Rio de Janeiro (2009)
19. Stargate gateway (spb400), http://www.xbow.com
20. Zeinalipour-yazti, D., Lin, S., Kalogeraki, V., Gunopulos, D., Najjar, W.A.: Microhash: An efficient index structure for flash-based sensor devices. In: Proceedings of the 4th Conference on File and Storage Technologies, FAST 2005, pp. 31–44. USENIX, Newcastle upon Tyne (2005)
21. Zhang, Y., Shi, J., Liu, Y., Zhang, Y.: Verifiable Fine-Grained Top-k Queries in Tiered Sensor Networks. In: Proceedings of the 29th IEEE International Conference on Computer Communications, INFOCOM 2010, pp. 1–9. IEEE, California (2010)