

# Poster: Secure and Efficient Query Processing in Sensor Networks

Fei Chen      Alex X. Liu  
 Dept. of Computer Science and Engineering  
 Michigan State University  
 East Lansing, MI 48824-1266, U.S.A.  
 Email: {feichen, alexliu}@cse.msu.edu

## I. INTRODUCTION

The architecture of two-tiered sensor networks (illustrated in Fig. 1), where storage nodes serve as an intermediate tier between sensors and a sink for storing data and processing queries, has been widely adopted because of power and storage saving for sensors as well as the efficiency of query processing. However, a compromised storage node imposes significant threats. First, it may allow attackers to obtain sensitive data stored in the storage node. Second, it may return forged data for a query. Third, it may not return all data items that satisfy the query. Several privacy and integrity preserving protocols [1], [2] have been proposed to prevent attackers from gaining information from both sensor collected data and sink issued queries<sup>1</sup>, and allows the sink to detect compromised storage nodes when they misbehave. However, the state-of-the-art protocol [1] has two main drawbacks: (1) it allows attackers to obtain a reasonable estimation on both sensor collected data and sink issued queries; (2) the power consumption and storage space for both sensors and storage nodes grow exponentially with the number of dimensions of collected data.

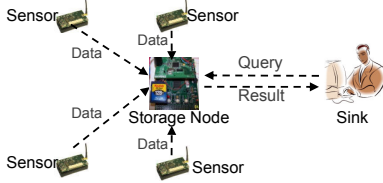


Fig. 1. Architecture of two-tiered sensor networks

In this paper, we propose SafeQ, a novel privacy and integrity preserving protocol for two-tiered sensor networks. To preserve privacy, SafeQ encodes both data and queries such that a storage node can correctly process encoded queries over encoded data without knowing their values. To preserve integrity, we propose the neighborhood chaining technique that allows a sink to verify whether the query result contains exactly the data items that satisfy the query. We also propose an optimization technique using Bloom filters to significantly reduce communication cost between sensors and storage nodes. Furthermore, we propose a solution to adapt SafeQ for event-driven sensor networks, where a sensor submits data when a certain event happens. Comparing with the state-of-the-art, SafeQ not only prevents attackers from knowing both sensor collected data and sink issued queries, but also delivers orders of magnitude better performance on both power consumption and storage space for multi-dimensional data.

<sup>1</sup>Queries in sensor networks typically can be modeled as range queries.

## II. PRIVACY FOR 1-DIMENSIONAL DATA

To preserve privacy, each sensor  $s_i$  encrypts data items  $d_1, \dots, d_n$  using its secret key  $k_i$ , denoted as  $(d_1)_{k_i}, \dots, (d_n)_{k_i}$ . Note that,  $k_i$  is a shared secret key with the sink. However, the key challenge is how a storage node processes encrypted queries over encrypted data without knowing their values. The idea of our solution is to convert sensor collected data and sink issued queries to prefixes, and then use prefix membership verification to check whether a data item satisfies a range query. To prevent a storage node from knowing the actual values of data items and range queries, sensors and the sink apply Hash Message Authentication Code (HMAC) to each prefix converted from the data items and range queries. For example, consider sensor collected data  $\{1, 4, 5, 7, 9\}$  and a sink issued query  $[3, 6]$  in Fig. 2. The sensor first converts the collected data to ranges  $[min, 1], [1, 4], \dots, [9, max]$ , where  $min$  and  $max$  denote the lower and upper bound for all possible data items, respectively. Second, the sensor converts each range  $[d_j, d_{j+1}]$  to prefixes, denoted as  $p([d_j, d_{j+1}])$ , and then applies HMAC to each prefix in  $p([d_j, d_{j+1}])$ , denoted as  $h_g(p([d_j, d_{j+1}]))$ . Third, the sensor sends the result to a storage node. When the sink performs query  $[3, 6]$ , it first converts 3 and 6 to prefixes, denoted as  $p(3)$  and  $p(6)$ , respectively, and then applies HMAC to each prefix in  $p(3)$  and  $p(6)$ , denoted as  $h_g(p(3))$  and  $h_g(p(6))$ , respectively. Upon receiving query  $h_g(p(3))$  and  $h_g(p(6))$  from the sink, the storage node checks which  $h_g(p([d_j, d_{j+1}]))$  has common elements with  $h_g(p(3))$  or  $h_g(p(6))$ . Based on prefix membership verification, if  $h_g(p(a)) \cap h_g(p([d_j, d_{j+1}])) \neq \emptyset$ ,  $a \in [d_j, d_{j+1}]$ . Therefore,  $h_g(p(3)) \cap h_g(p([1, 4])) \neq \emptyset$  and  $h_g(p(6)) \cap h_g(p([5, 7])) \neq \emptyset$ . Finally, the storage node finds that the query result of  $[3, 6]$  includes two data items 4 and 5, and then sends  $(4)_{k_i}$  and  $(5)_{k_i}$  to the sink.

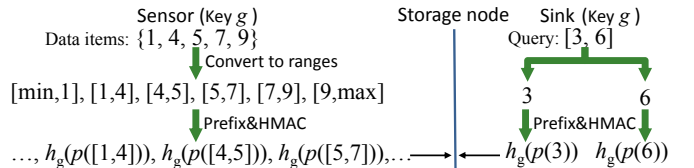


Fig. 2. Privacy preserving scheme of SafeQ

## III. INTEGRITY FOR 1-DIMENSIONAL DATA

To allow the sink to verify the integrity of a query result, the query response from a storage node to the sink consists of two parts: (1) the *query result*  $QR$ , which includes all the encrypted data items that satisfy the query; (2) the *verification object*  $VO$ , which includes information for the sink to verify the

integrity of  $QR$ . We present *neighborhood chaining* technique to preserve integrity of a query result. The idea is that instead of encrypting each data item individually, a sensor encrypts each item with its left neighbor such that if a storage node excludes any data item that satisfies the query, the sink can detect it. Fig. 3 shows the neighborhood chain for the sensor collected data in Fig. 2, where “|” denotes concatenation. For the range query [3,6], the query result  $QR$  is  $\{(1|4)_{k_i}, (4|5)_{k_i}\}$  and the verification object  $VO$  is  $\{(5|7)_{k_i}\}$ . If a storage node excludes  $(4|5)_{k_i}$  in  $QR$ , the sink can detect this error because the items in  $QR$  and  $VO$  do not form a neighborhood chain.

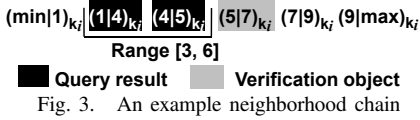


Fig. 3. An example neighborhood chain

#### IV. QUERIES OVER MULTI-DIMENSIONAL DATA

To preserve the privacy of multi-dimensional data, we apply our 1-dimensional privacy preserving techniques to each dimension of multi-dimensional data. For example, considering 5 two-dimensional data items (1,11), (3,5), (6,8), (7,1) and (9,4), sensor  $s_i$  applies the 1-dimensional privacy preserving techniques to the first dimensional values  $\{1, 3, 6, 7, 9\}$  and the second dimensional values  $\{1, 4, 5, 8, 11\}$ . Given a range query  $([2,6],[3,8])$ , the query result  $QR^1$  for the sub-query [2,6] consists of the encrypted data items of (3,5),(6,8) and the query result  $QR^2$  for the sub-query [3,8] consists of the encrypted data items of (9,4),(3,5),(6,8). Therefore, the query result  $QR$  consists of the encrypted data items of (3,5),(6,8).

To preserve the integrity of multi-dimensional data, we build a multi-dimensional neighborhood chain. The idea is that for the value of each dimension in a data item, we find its left neighbor along each dimension and embed this information when we encrypt the item. Such neighborhood information is used by the sink for integrity verification. Considering 5 example 2-dimensional data items (1,11), (3,5), (6,8), (7,1), (9,4) with lower bound (0,0) and upper bound (15,15), the corresponding multi-dimensional neighborhood chain encrypted with key  $k_i$  is  $(0|1, 9|11)_{k_i}$ ,  $(1|3, 4|5)_{k_i}$ ,  $(3|6, 5|8)_{k_i}$ ,  $(6|7, 0|1)_{k_i}$ ,  $(7|9, 1|4)_{k_i}$  and  $(9|15, 11|15)_{k_i}$ . Fig. 4 illustrates this chain, where each black point denotes an item, the two grey points denote the lower and upper bounds, the solid arrows illustrate the chain along the X dimension, and the dashed arrows illustrate the chain along the Y dimension.

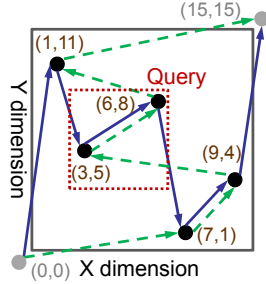


Fig. 4. A 2-dimensional neighborhood chain

#### V. SAFEQ OPTIMIZATION

To reduce the communication cost between sensors and storage nodes, for  $n$  data items  $d_1, \dots, d_n$ , we use a Bloom filter to represent  $h_g(p([\min, d_1]))$ ,  $h_g(p([d_1, d_2]))$ ,  $\dots$ ,  $h_g(p([d_{n-1}, d_n]))$ ,  $h_g(p([d_n, \max]))$ . Thus, a sensor only needs to send the Bloom filter instead of the hashes to a storage node. The number of bits needed to represent the Bloom filter is much smaller than that needed to represent the hashes. Taking  $h_g(p([4, 5]))$  and  $h_g(p([5, 7]))$  in Fig. 2 as the example, we assume  $h_g(p([4, 5])) = \{v_1\}$  and  $h_g(p([5, 7])) = \{v_2, v_3\}$ .

$h_g(p([4, 5]))$  and  $h_g(p([5, 7]))$  can be represented as the two arrays in Fig. 5, where  $A$  is a bit array representing the Bloom filter and  $B$  is an array of pointers. Each pointer points to a list of indexes of ranges, e.g., 2 is the index of [4,5] and 3 is the index of [5,7]. Note that, “-” denotes a null pointer.

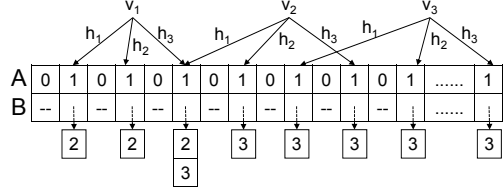


Fig. 5. An example Bloom filter

#### VI. RANGE QUERIES IN EVENT-DRIVEN NETWORKS

So far we have assumed that at each time slot, a sensor sends to a storage node the data that it collected at that time slot. However, this assumption does not hold for event-driven networks, where a sensor only reports data to a storage node when certain event happens. If we directly apply our solution here, the sink cannot verify whether a sensor collected data at a time slot. We propose the *idle period* technique to address this challenge. The idea is that a sensor reports its idle period to a storage node when it submits data after an idle period or when the idle period is longer than a threshold. The storage node can use the idle period to prove to the sink that the sensor did not submit any data at any time slot in that idle period. Fig. 6 illustrates two idle periods  $[t_1, t_2]_{k_i}$  and  $[t_3, t_4]_{k_i}$ , where each unit in the time axis is a time slot, a grey unit denotes that  $s_i$  has data to submit, a blank unit denotes that  $s_i$  has no data to submit, and  $\gamma$  is the threshold.

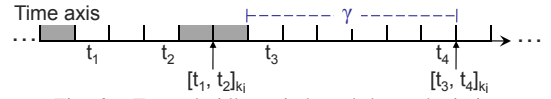
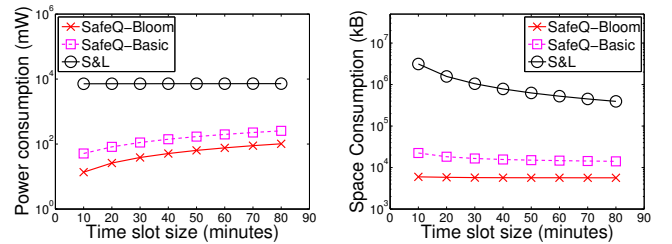


Fig. 6. Example idle periods and data submissions

#### VII. EXPERIMENTAL RESULTS

We implemented both SafeQ and the state-of-the-art (represented by S&L scheme) on a large real data set [3]. For 3-dimensional data, in comparison with S&L scheme, our experimental results show that, SafeQ-Bloom consumes 184.9 times less power for sensors and 182.4 times less space for storage nodes; SafeQ-Basic consumes 59.2 times less power for sensors and 58.5 times less space for storage nodes. Fig. 7 shows the average power and space consumption for 3-dimensional data.



(a) Sensor: power consumption (b) Storage node: space consumption  
Fig. 7. Ave. power and space consumption for 3-dimensional data

#### REFERENCES

- [1] B. Sheng and Q. Li, “Verifiable privacy-preserving range query in two-tiered sensor networks,” in *Proc. IEEE INFOCOM*, 2008, pp. 46–50.
- [2] J. Shi, R. Zhang, and Y. Zhang, “Secure range queries in tiered sensor networks,” in *Proc. IEEE INFOCOM*, 2009.
- [3] “Intel lab data,” <http://berkeley.intel-research.net/labdata>.