


Nasser Kimbugwe^{1,2} and Tingrui Pei^{1,3*} ¹Xiangtan University, Xiangtan China;²Makerere University, Kampala, Uganda³Key Laboratory of Internet of Things and Information Security of Hunan province, Education Ministry, Xiangtan, China

*e-mail: peitingrui@xtu.edu.cn

An Efficient Privacy and Integrity-Preserving Range Query Scheme over Unsorted Data in IoT Two-tiered Wireless Sensor Networks

Abstract. In most existing privacy-preserving range-query schemes, sorting data items is extremely critical for enabling integrity verification over query results. In two-tiered WSNs, the data is collected by several sensors and send to a central node called the storage node. If data is not sorted, it is hard for the sink to verify query result integrity without sending all data items to the sink. To preserve data privacy on the storage node, the sensors must do sorting, and data is encrypted before it's sent to the storage nodes since storage nodes are more vulnerable to adversary attacks than the sensor nodes. However, sensors are resource-constrained, yet the sorting process is resource-consuming. The higher the number of data items collected, the larger the number of resources required to sort the elements. In this paper, on top of proposing a scheme that assumes that the data is sorted, we also propose a scheme we call Probabilistic Verification Neighborhood chaining (PvNC) for both sorted and unsorted data that still preserves data and range-query privacy and to a great extent guarantee data integrity. Experimental results show that sensor nodes use less power using our PvNC for unsorted data than the existing schemes that require data to be sorted.

Key words: two-tiered sensor networks; range query; privacy and integrity preserving; unsorted data.

Introduction

Wireless Sensor Networks (WSNs) have been widely deployed in different settings ranging from temperature monitoring, fire outbreak sensing, building safety monitoring, and underwater surveillance among others. WSNs are indeed indispensable building blocks for the Internet of Things functionalities. Originally, each sensor was required to send real-time data to the network owner (Sink), which proved costly due to unreliable and cost ineffective communication links. This gave rise to the Two-Tiered WSNs [1], where the lower layer consists of resource-constrained sensor nodes(s_i) while the upper layer consists of resource-rich Storage Nodes(\mathcal{M}). s_i collects data periodically and sends it to the closest \mathcal{M} for storage and query processing. The two-tiered WSNs were enhanced by the invention of powerful sensors with more storage and processing power [2].

A range query is one of the rudimentary operations done over encrypted data especially in Wireless Sensor Networks (WSN) and cloud computing setting. Range queries are in a form $[a,b]$ where the processing entity should return all the data that falls in range $[a b]$ if any. The other common version of range queries is the Top-k query where the

processing entity returns all the values above or below k [3] and MIN/MAX queries [4]. In two-tiered WSN, the processing entity is the central storage node (\mathcal{M}) and it is where all the range query requests are directed.

Due to the importance of \mathcal{M} in two-tiered WSNs, they (\mathcal{M}) are always a target for adversaries. Once compromised, the adversary can gain control over \mathcal{M} and this may violate data privacy and integrity. However, most adversaries want to control the storage node without being detected but can cause serious damage like changing or dropping vital results from the range query upon which the owner can base on to take wrong decisions thinking that it is authentic data. This, therefore, requires two urgent solutions: 1) Being able to protect and preserve the privacy of the data collected by the sensor nodes while permitting queries to be carried out over that data, and 2) Being able to verify the integrity of the range query results from \mathcal{M} to be sure that:-

- i) The data returned by \mathcal{M} satisfies the range query,
- ii) The returned range query results are valid and not merely forged,
- iii) No qualifying data has been dropped from the range query results.

As a result, intensive research has been carried out to tackle these problems and several schemes have been proposed [3], [5]–[8] among others. Most of these approaches [3], [5]–[8], however, assume that the data collected by the sensor nodes is in sorted form or else should be first sorted. In real world, data is in unsorted form and sorting it places extra task on the sensor nodes whose resources are already limited. Most commonly used sorting algorithms are on average $O(N^2)$ [9] for example bubble and selection sort [10], it is therefore a huge task if sensors have to sort huge amounts of data, yet their processing power is limited. Even with more advanced sorting algorithms that are mostly $O(n \log n)$, sorting data is still an expensive operation for resource-constrained sensors.

Another justification for need to devise privacy-preserving range query schemes over unsorted data is that most wireless sensor networks are deployed in very harsh environments where they are vulnerable to attacks. Even if data is sorted on the fly, the sensors must wait for the last data collected to confirm that all the data is sorted. Having to wait for the end of each timeslot to sort the data is risky since the adversary can have access to relatively significant amount of data within a single timeslot in case of an attack. To reduce on this risk, data has to be encrypted on the fly. However, sorting encrypted data without revealing significant amount of information about data being sorted is challenging. Schemes for unsorted data are even more significant in cases where data encrypted and sent to the storage node in real-time. Because data is already encrypted, sorting it on the storage node before range query can be carried out is still a big challenge. Moreover, storage nodes are more vulnerable to attacks than the sensor nodes and therefore sending unencrypted real-time data to be sorted on the storage node side is not a good strategy.

The contributions of this paper are summarized as follows:

- We first propose a privacy-preserving and integrity verification scheme called Adjacency-difference verification scheme (AVC) which, like the previous studies, assumes that the data items collected by the sensor are first sorted at the sensor side before being sent to the storage node. AVC uses 0-1 encoding and Hash-based Message Authentication Code (HMAC) functions to both protect the data and to carry out range query search over the encrypted data. The adjacency difference of the data elements helps in range query integrity verification.
- We propose a privacy-preserving and integrity verification scheme for unsorted data called

Probabilistic Verification Neighborhood chaining (PvNC) which relieves the sensors the burden of sorting data.

- We carry out both theoretical and experimental analysis of our proposed schemes. Our experiments show that our AVC scheme has favorable communication cost. PvNC achieve privacy-preserving range query processing over unsorted data with less power consumption by sensor nodes during the initial submission stage compared to the existing schemes.

The rest of the paper is organized as follows. In Section 2 we present related work. In section 3 we discuss the system and threat models. In section 4 we discuss our range query processing model for sorted data. Section 5 presents the proposed PvNC scheme for unsorted data. In section 6 we discuss the performance evaluation and the experiments of our schemes. Finally, the conclusion is drawn in Section 7.

Review of related work

In recent years, research efforts have focused on exploring and improving the privacy-preserving range queries as well as being able to verify the integrity of range query results.

Various techniques for preserving privacy and integrity of range queries over data in WSNs have been proposed. In [6] the authors adopted the data partitioning technique in [11] known as the bucketization technique for preserving data and range query privacy by attaching tags to each bucket. In this scheme, data is assumed to be in sorted order or the sensor node has to sort it during the bucket assigning process.

To verify range query integrity, B. Sheng and Q. Li[6] use coding bits technique. Their technique works in a way that, if a sensor does not have any data that falls into particular buckets, it generates and appends coding bits to the data it sends to the storage node for every bucket where it does not have any data during epoch t . When \mathcal{M} claims that there is no data belonging to particular buckets during epoch t , it has to send the coding bits to the sink as proof. However, this approach will generate too many coding bits when applied to sensing activities where the expected data will most likely fall into fewer buckets. For example, temperature sensing where the temperature remains constant for a long time.

A spatiotemporal approach was suggested by J. Shi et al. in [12] which requires every sensor to broadcast some information about the data it has collected in every epoch t . This information includes

the ID of the sensor and ‘index data’ that indicates the buckets for which sensor s_i has some data or not. It also works only if the data is in sorted order. Every sensor then has to embed all other sensors’ index data into its data packet before it sends its packet to the Storage node. This creates unnecessary overhead because of duplicated index data sent and stored on the Storage node.

J.Zeng et al.[13] proposed a privacy-preserving, energy-efficient and multi-dimensional range query protocol called PERQ which provides efficient privacy-preserving but also requires each sensor to include some ‘data integrity verification information’ in each of its collected data before sending it to the storage node. They also suggest a cyclic modular verification scheme for multi-dimension range queries but also require embedment of verification information in each of the collected data item. We argue that this still creates unnecessary index data overload, which may in a long run constrain the storage node. In addition, it also works only when the data is sorted.

Chen and Liu [14] used Merkle hash tree to build a verification object used for range query verification. They also assume that the data is sorted in ascending order and the verification object

contains the value before the lowest range query result element and the value after the range query largest element. Rui Li et al[3] proposed a scheme for efficient top-k query processing in WSNs. In their scheme, they first transform arbitrarily distributed data into a uniform distribution to enable top-k query processing on the data.

X. Zhang et al[15] proposed the first collusion aware privacy-preserving and integrity verification scheme in two-tiered WSNs. In their scheme, each data item’s position is concatenated with the element in that position. To verify the integrity of the range query results, the positions of the received elements must be sequential. This is only possible when elements are sorted which is not always the case. Their model also assumes that the sensor data is sorted or in case it’s not sorted, the sensor must first have to sort it before sending it to the storage node.

In[7], authors proposed a data structure called Encrypted constraint chain in which numbers stored in ascending order are partitioned into several parts with a given parameter, encrypted separately and can be brought together by concatenating the different parts together. This model can only work if the elements are sorted, but that is not always the case in the real world.

Table 1 – Notations

Symbol	Meaning	Symbol	Meaning
s_i	Sensor node i	$C_{s_i \rightarrow \mathcal{M}}$	Communication cost for sending data from s_i to \mathcal{M}
d_i	Data item collected by sensor i	f_d	The size of data item d_i in bits
ℓ	Number of elements chained with d_i	f_{id}	The size of sensor id in bits
$\ell_{d_i, left}$	All d_i ’s left side elements	f_t	The size of time slot t in bits
$\ell_{d_i, right}$	All d_i ’s right side elements	f_H	The HMAC encoding of data items’ 0-1 encodings
$E^{0-1}(d_j)$	0-1 encoding of d_j	$E^1(d_j)$	1-encoding of d_j
$E^0(d_j)$	0-encoding of d_j	Q	Number of sensors in a cell
K_{it}	Encryption key for sensor i during time t .	n	Number of data items collected by a sensor
q_i	Key shared between s_i and the sink	N	Network size

Models and problem statement

In this section, we discuss the models used in our study. In 3.1, we discuss the system model, which is the two-tiered wireless sensor network model. In 3.2 we elaborate on the adversary model and we finally discuss the range query model in 3.3.

System Model

We adopt the Two-Tiered Wireless Sensor Networks model [1] as shown in Figure 1.

Consisting of storage nodes and normal sensors. We assume the network is divided into several cells, each cell with several sensors and a storage node. Sensors in a cell send their collected data to the storage node at the end of every stipulated time slot t . The Sink in this case represents the network owner or any other authorized network user.

The storage nodes play two important roles in two-tiered WSNs. 1) They permanently store the data submitted by the various sensor nodes and 2) They

process the queries from the sink. This saves resources that each individual sensor would be required to have. When the sink requires any data collected by the sensors, it sends the query directly to

the storage node instead of the sensor node. This improves query-processing speed since the storage nodes have more processing power than the sensor nodes.

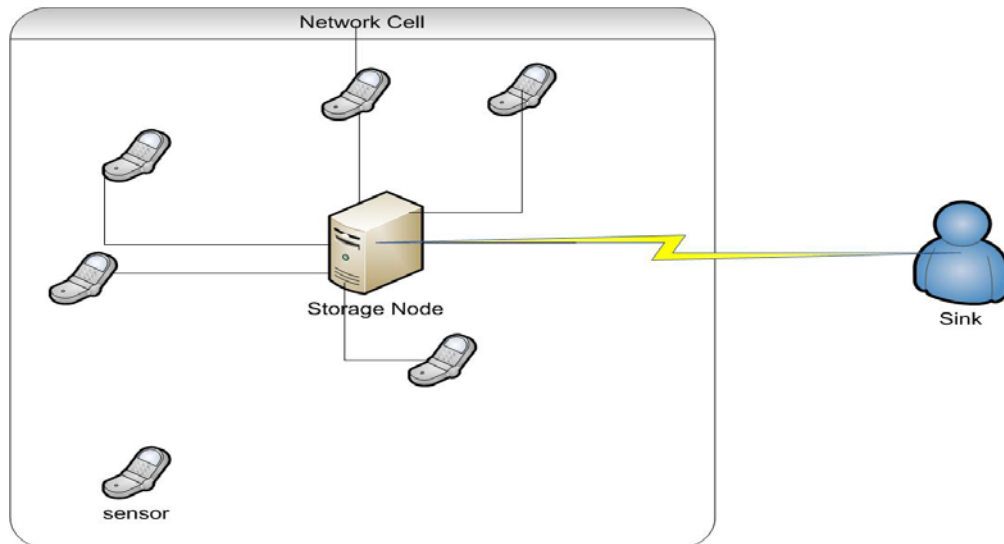


Figure 1 – Two-Tiered WSN model consisting of the storage node and different sensors

Adversary Models

Our focus is ensuring the privacy of both the stored data and the range query as well as the integrity of the range query results in two-tiered WSNs setting. Because each sensor has significantly fewer data compared to the storage node, it is less likely that the adversary will target individual sensor nodes. As in [14], we assume that normal sensors are trusted and our focus is on storage nodes. However, in the unlikely event of an attack on normal nodes, the damage is minimal given the amount of data each sensor holds compared to the storage node. If the adversary gains control of the storage node, he may influence the results of the range query but in most case in a malicious way. Since his aim is to remain obscure, carrying out obvious actions like completely deleting the data may not be an option for him. Mainly, he can do any of the following: 1) Forge the query results.

However, since each data item is encrypted using a unique key shared only between the sensor and the sink, any forged data can easily be detected unless the adversary can successfully guess or compute the encryption key, which is computationally infeasible. 2) Include data that is outside the query range. The sink can easily detect this by decrypting the data and compare it with the range query. 3) The storage node

can intentionally drop some data that satisfies the query, which leads to incomplete range query results. This is difficult to detect and it's the focus of this paper.

Range Query Model

A range query requires access to all the data items that are within a specific range $[a, b]$, where a is the lower boundary and b is the upper boundary of the elements requested for. In other words, no element smaller than a or larger than b should be included in the range query results. A range query for sensor-collected data is in the following form $Q = \{s_i, t, [a, b]\}$, where s_i and t are sensor node id and time slot respectively. Note that more than one sensor s_i can be specified in a single query. For brevity, we assume that the range query contains a single s_i . We also assume that the query is directed to a single storage node within a given network cell. Our model can be extended to cater for multiple sensors and cells by merely merging the range query results from the different cells and sensors.

Adjacency-difference verification scheme (avc) for sorted data

Like most previous studies [3]–[8], [15]–[18], we also first assume that data items collected by the

sensors are in sorted order and proposed a protocol for privacy-preserving and integrity verification in WSNs.

For brevity, we only consider sensors within a single cell, much as a network can have multiple cells each with a storage node. However, our approach can easily be extended to the entire network. At the end of every time slot, every sensor s_i collects data elements and sorts them in ascending order i.e $D = \{d_1, d_2, d_3, \dots, d_n\}$, where $d_1 < d_2 < \dots < d_n$

Our idea, just like in [14] and [19], is to embed some relationship information in each of D_i 's data items that can help us verify whether the data elements in D_i are adjacent neighbors or not after

$$AVC = \{(d_1 || d_1 - d_{min})k_{it}, (d_2 || d_2 - d_1)k_{it}, \dots, (d_n || d_n - d_{n-1})k_{it}, (d_{max} || d_{max} - d_n)k_{it}\}$$

Our scheme assumes that the order of the elements is maintained when submitted to the storage node. The AVC is very vital in query authenticity and integrity verification. Our approach is closely related to the neighborhood chain in [14] and [19]. However, in [14], their approach combines two adjacent data element that is $\{(d_1 || d_2)k_{it}, (d_2 || d_3)k_{it}, \dots, (d_{n-1} || d_n)k_{it}\}$ and requires a verification Object(OV) to be sent to the sink to enable the sink verify the query integrity. In [19], they embed both right and left neighbor of each element which increases the communication cost.

As another way of verifying the integrity of the range query in our approach, when the sensor does not collect any data during a particular time slot t , its required to send AVC data structure that only contains systems dummy values d_{min} and d_{max} which should be set somehow below the minimum expected value and maximum expected value such that at no one time, should $d_j \leq d_{min}$ or $d_j \geq d_{max}$.

AVC Secure Range Query Processing Protocol

Secure range query protocols are concerned with preventing malicious storage node from inferring meaningful information from both the query and the query results that can in one way or another reveal the actual contents of the data stored on the storage node.

We adopt the 0-1 encoding mechanism that was defined in [20] in solving the millionaires' problem. Using 0-1 encoding mechanism, it is possible to determine whether two numbers are equal or not without using the equality ($=$), less than ($<$) or greater than ($>$) operators. To determine whether a

number is within a range, it is sufficient to determine if there is intersection between two sets of corresponding 0-1 encoding. Given a number L , we first computer its binary equivalent in form $\{b_1 b_2 \dots b_{n-1} b_n \in \{0,1\}^n\}$, where n is the bit length of the number L . The 0-encoding of L represented as $E^0(L) = \{b_i b_{i+1} b_{i+2} \dots b_n | b_i = 0 \wedge 1 \leq i \leq n$ and the 1-encoding of L is represented as $E^1(L) = \{b_i b_{i+1} b_{i+2} \dots b_n | b_i = 1 \wedge 1 \leq i \leq n\}$. If $E^1(L) \cap E^0(\lambda) \neq \emptyset$, then *definitely* $L > \lambda$, else $L \leq \lambda$.

We call this an *Adjacency-difference Verification Chain-scheme (AVC)*.
After sorting the data element, the sensor constructs the AVC of each element by concatenating each data element with the difference between itself and its immediate left position element. We assume that the first element collected by the sensor takes position 1, the second takes position 2 and so on up to position n , where n is the number of the data elements collected by the sensors during t . The elements in position 0 and position $n+1$ are the domain minimum and maximum values which are well known to each sensor within the network. We call them d_{min} and d_{max} respectively.

We transform each 0-1 encoding to a unique number by adopting the enumeration function mechanism as in [14] and [7] and then secure it using one-way keyed-Hash Message Authentication Code (HMAC) functions. By using HMAC functions, it becomes infeasible for the storage to try to steal sensitive information from both the data sent by the sensors and the queries sent by the sink.

We denote the HMAC encoded 0-1 encoding of a number L after applying HMAC function on it as $NE^{0-1}(L) = HMAC_{q_i}(\check{N}(E^{0-1}(L)))$ where \check{N} is a numerical function, and q_i is a key form HMAC shared only between the sensor node s_i and the sink.

From the above denotation, $NE^1(L) = HMAC_{q_i}(\check{N}(E^1(L)))$ and $NE^0(L) = HMAC_{q_i}(\check{N}(E^0(L)))$. To find out if a number L is greater than λ , $HMAC_{q_i}(\check{N}(E^1(L))) \cap HMAC_{q_i}(\check{N}(E^0(\lambda))) \neq \emptyset$.

AVC Submission Protocol

This section describes the manipulations that the sensor performs on the data elements before submitting the data to the nearby storage node. Let

$D_i = \{d_1, d_2, d_3, \dots, d_n\}$, where $d_1 < d_2 < \dots < d_n$ be the data collected by sensor s_i during time slot t . Before submitting the data to the nearest storage node M , the sensor performs the following tasks:

(2)

$$E^{0-1}(d_j) = \{E^1(d_j), E^0(d_j)\} \tag{1}$$

$$\text{Let } \hat{E}_i = \{E^{0-1}(d_{min}), E^{0-1}(d_1), E^{0-1}(d_2), \dots, E^{0-1}(d_{n-1}), E^{0-1}(d_n), E^{0-1}(d_{max})\} \tag{2}$$

(3) Enumerate each 0-1 encoding using a numerical function $\mathcal{N}(\ast)$ and encode the enumerated value using HMAC function to prevent the storage from stealing valuable information. We call this enumerated coded numbers (ECN). We define ECN as

$$ECN_i = HMAC_{q_i}(\mathcal{N}(\hat{E}_i)) \tag{3}$$

where q_i is an encoding secret key only shared between the sensor S_i and the sink.

(4) Compute the

$$AVC_i = \{(d_1 || d_1 - d_{min})k_{it}, (d_2 || d_2 - d_1)k_{it}, \dots, (d_n || d_n - d_{n-1})k_{it}, (d_{max} || d_{max} - d_n)k_{it}\} \tag{4}$$

(5) Submit the following to the nearest storage node.

$$S_i \rightarrow \mathcal{M}: i, t\{AVC_i, ECN_i\} \tag{5}$$

(6) If s_i does not collect any data during a particular time slot t , the sensor only sends the system minimum and maximum denoted as d_{min} and d_{max} . In our approach, we suggest these to be far smaller or far bigger than the lowest or highest value the sensor can ever collect and should only be known by the sensors and the sink.

Therefore, in this case, the sensor sends $s_i \rightarrow \mathcal{M}: i, t, \{(d_{min})_{kit}, (d_{max})_{kit}, ECN_{min}, ECN_{max}\}$.

This will act as a secret check when the storage claims that no data was submitted by the sensor during a particular time slot t . Without the data encryption key and the HMAC function key, it's computationally infeasible for the storage node to steal sensitive information data submitted by the sensor.

Example:

We now illustrate our approach using an example. Let $\mathcal{D}_i = \{3, 7, 13, 23, 26, 38\}$ be a set of elements collected by s_i during time slot t . Let d_{min} and d_{max} be 0 and 100 respectively.

$$AVC_i = \{(3||3-0)(7||7-3)(13||13-7)(23||23-13)(26||26-23)(38||38-26)(100||100-36)\}$$

$$AVC_i = \{(3||3)k_{it}, (7||4)k_{it}, (13||6)k_{it}, (23||10)k_{it}, (26||3)k_{it}, (38||8)k_{it}, (100||64)k_{it}\}$$

$$\hat{E}_i = \{E^{0-1}(0), E^{0-1}(7), E^{0-1}(13), E^{0-1}(23), E^{0-1}(26), E^{0-1}(38), E^{0-1}(100)\}$$

$$ECN_i = \{HMAC_q(\mathcal{N}(\hat{E}_i))\}. \text{ So the sensor sends } s_i \rightarrow \mathcal{M}: i, t, \{AVC_i, ECN_i\}$$

AVC Range Query Submission and Processing Protocol

The range query protocol is concerned with how to protect query processing and range query results' privacy. If the storage node can successfully steal sensitive information by studying the range query, then the original data

encryption by the sensors becomes useless. It is also very important for the sink to be able to verify the authenticity and integrity of the query in order to be able to detect malicious storage nodes that might manipulate the range query results. Let $Q_R = \{T, [a, b]\}$ be a range query where T is a set of time slots and $[a, b]$ represents the minimum and the

maximum values that should be included in the range query results respectively. For every value between a and b, the sink computes its 0-1 encoding, enumerates it using $\mathcal{N}(\ast)$ and finally

$$Sink \rightarrow \mathcal{M}: i, \{T, [HMAC_{q_i}(\mathcal{N}(E^{0-1}(a))), HMAC_{q_i}(\mathcal{N}(E^{0-1}(b)))]\} \tag{6}$$

where $E^{0-1}(a) = \{(E^1(a), (E^0(a))\}$ and $E^{0-1}(b) = \{(E^1(b), (E^0(b))\}$ respectively.

Upon receiving the range query from the sink, the storage node processes the query and sends the query results back to the sink. The following steps are

$$\begin{aligned} & (HMAC_{q_i}(\mathcal{N}(E^1(d_j))) \cap HMAC_{q_i}(\mathcal{N}(E^0(a))) \neq \emptyset) \\ & \wedge (HMAC_{q_i}(\mathcal{N}(E^1(d_j))) \cap HMAC_{q_i}(\mathcal{N}(E^0(b))) = \emptyset) \end{aligned}$$

Proof:

From [20], if $E^1(x) \cap E^0(y) \neq \emptyset$, then $x > y$. From this definition therefore, if $(HMAC_{q_i}(\mathcal{N}(E^1(d_j))) \cap HMAC_{q_i}(\mathcal{N}(E^0(a))) \neq \emptyset$, it means that $d_j > a$ and if $(HMAC_{q_i}(\mathcal{N}(E^1(d_j))) \cap HMAC_{q_i}(\mathcal{N}(E^0(b))) = \emptyset$

It means that $d_j \leq b$ which basically means that d_j falls in the range $[a, b]$ i.e $d_j \in [a, b]$.

(2) The storage finds all the encrypted AVC data elements that match the query if any. Let $AVC_{query} \subseteq AVC$ be a set of all AVC elements that satisfy the query. As a way of verifying the range query results, the storage node must include d_{r+1} where r is the number of elements in that satisfy the query, and d_r is the largest element in AVC_{query} . If the largest element in AVC_{query} is the last element in, then d_{max} must also be included in AVC_{query} . The storage sends the following message to the sink

$$\mathcal{M} \rightarrow Sink: i, \{AVC_{query}\}$$

(3) In case \mathcal{M} finds no data that satisfies Q_R , it is required to send d_{min} and d_{max} together with their corresponding encoded 0-1 codes $HMAC_{q_i}(\mathcal{N}(E^{0-1}(d_{min})))$ and $HMAC_{q_i}(\mathcal{N}(E^{0-1}(d_{max})))$. This is the only time the storage is required to include d_{min} in the range query results.

Upon receiving the range query result from \mathcal{M} , the sink takes the following steps in order to extract the actual values that satisfy the query and to verify the authenticity and integrity of the range query.

encodes it using $HMAC_q(\ast)$ where q is the key shared between the sink and the sensor. A union set of all the coded 0-1 values is constructed. The final query sent to the storage node by the sink is

undertaken by the storage node to process the query results.

(1) For every data element with time slot included in T, the storage node compares its HMAC encoded 1-encoding with the union of a and b HMAC encoded 0-encoding. An element $d_j \in [a, b]$ if :

i. Decrypts all the AVC data elements within the range query and extract the values of the query. For example, if the query range was [5,30] and the query result has $AVC_{query} = \{(7||4)k_{it}, (13||6)k_{it}, (23||10)k_{it}, (26||3)k_{it}, (38||8)k_{it}\}$ the sink decrypts the AVC_{query} to get $\{(7||4), (13||6), (23||10), (26||3), (38||8)\}$. The first value in every data element before the concatenation is the actual range query data, while every value after the concatenation is the difference between the current value and the data value before it. The last element in AVC_{query} is the results upper boundary and is only required for query verification purposes. Therefore, the actual range query values are $\{7, 13, 23, 26\}$

ii. Then verifies the integrity of the query. The verification method depends on whether $AVC_{query} = \emptyset$ or not. If $AVC_{query} \neq \emptyset$, given that $AVC_{query} = \{d_1||p_1, d_2||p_2; \dots, d_{r-1}||p_{r-1}, d_r||p_r\}$, where r is the total number of elements in AVC_{query} and $p_r = d_{r+1} - d_r$, the sink verifies the query in the following ways.

- $d_r \notin [a, b]$ i.e the last element of the query result must be outside the query range
- For $\geq 2, d_r - d_{r-1} = p_{r-1}$. This condition is sufficient to check for either deletion of data from the range query results or addition of data forged into the range query results that is within the range. If \mathcal{M} deletes some data (apart from the first element), the adjacency-difference between neighboring elements will be wrong. Similarly, if \mathcal{M} inserts some data

within the range query results, it will also have to guess the adjacency-difference in order to escape being noticed. But since the AVC is encrypted, \mathcal{M} has to guess the right difference in order to escape being noticed.

- If \mathcal{M} deletes what would have been the first element of AVC_{query} , it can still be detected by examining the current first element. Let d_{del} be the first element that was deleted. If $(d_{del} = d_1 - p_1) \in [a, b]$, then \mathcal{M} can be suspected of being malicious since it eliminated data that should have been included in the range query results.

- If $AVC_{query} = \emptyset$, then there are two possibilities. 1) Either there is no data that satisfies the query but the data was indeed collected by the sensor s_i during time slot t or 2) the adversary has intentionally deleted all the data from the range query results. The first scenario is solved by always including the left boundary and the right boundary in the range query results. For example, if the query is $[a, b]$, we always include the biggest data value that is less than a as the left boundary and the smallest data value that is bigger than b as the right boundary of the range query results. For example, if s_i collects data $\{3, 7, 13, 23, 26, 38\}$, but the query is say $[8, 10]$, which means there is no data that falls in the range $[8, 10]$. Then the storage node should send $[7, 13]$ as the range query results. The sink can verify this by decrypting the received boundary data and then the adjacency-difference of the received data. In case of scenario 2, the sink will request the storage node to send d_{min} and d_{max} together with their corresponding encoded 0-1 codes $HMAC_q(\mathcal{N}(E^{0-1}(d_{min})))$ and $HMAC_q(\mathcal{N}(E^{0-1}(d_{max})))$ for the time slot t . Note that the only time s_i is required to send d_{min} to \mathcal{M} is when s_i collects no data at all during t . Therefore, if indeed data exists for time t , then d_{min} will not exist and the sink can suspect \mathcal{M} to be malicious.

AVC Communication Cost

Generally, the main concerns about range queries in WSNs are privacy of the data, and the communication cost of the models that preserve privacy while enabling search abilities on the encrypted data. Privacy-preserving and integrity checking have been discussed above in section 4.3. We consider communication cost at two levels; 1) the communication cost incurred by the sensor node to transmit data to the storage node and 2) The cost of submitting the range query by the sink to the storage node and the cost of transmitting range query result

together with the range query integrity verification information.

Communication Cost of Sensor s_i

We analyze the communication cost of our AVC scheme in bits. We consider a single cell with Q sensors (s_i) where each s_i collects n data items during time slot t . The communication cost for submitting data items from all sensors to the storage node within a cell is given by

$$C_{s_i \rightarrow \mathcal{M}} = \sum_{i=1}^Q \left[\left[\sum_{j=1}^n [(2f_d) + 2f_H] \right] + (f_{id} + f_t) \right] * \text{hop} \quad (7)$$

Where Q is the number of sensors in the cell, n is the number of items collected by each sensor during time t , hop is the number of average hops from each sensor to the storage node while f_d, f_{id}, f_t , and f_H are defined in Table 1.

Proof: In AVC, each data element is concatenated with another value equal to the difference between itself and the value before it. If the data item d_i had f_d bits, then $d_i - d_{i-1}$ must also have f_d bits. Therefore each data item's AVC is equal to $2 * f_d$. The sensor has to compute the HMAC values of each data item's 0-1 encoding. So if a sensor collects k data values, the total AVC and HMAC values will be $\sum_{i=1}^n (2f_d + 2f_H)$. Each sensor also submits its sensor id (f_{id}) and the time slot f_t during which the data was collected all added together to get $(f_{id} + f_t)$. All these summed together, each sensor submits $(\sum_{i=1}^n (2f_d + f_H)) + (f_{id} + f_t)$. Since each cell has n sensor, the total communication cost is the summation of all sensors' communication costs within the cell hence equation (3).

Communication Cost of Range Query processing

Range query communication cost involves two parts. One is about the query submission to \mathcal{M} by the sink and the second is concerned about \mathcal{M} range query results to the sink. From equation (6), the sink submits the sensor id(s), the time slot and the encrypted 0-1 encoded HMAC values of $[a, b]$. So communication cost of the query from the sink to \mathcal{M} denoted as $C_{sink \rightarrow \mathcal{M}} = f_{id} + 4f_H + f_t$.

$$C_{sink \rightarrow \mathcal{M}} = f_{id} + 4f_H + f_t. \quad (8)$$

For simplicity, we assume that range query is directed to data from a single sensor s_i and as such

we have a single f_{id} . We multiply f_H by 4 because each value has both 0-encoding and 1-encoding computed separately. So $[a, b]$ will have 4 HMAC encoded values hence equation (8).

If \mathcal{M} finds g items that satisfy the range query, then the cost of transmitting the query results to the sink is given by

$$C_{\mathcal{M} \rightarrow \text{sink}} = f_{id} + 2gf_d + f_t \tag{9}$$

$$C_{total} = C_{s_i \rightarrow \mathcal{M}} + C_{query} = \sum_{i=1}^Q \left[\left[\sum_{j=1}^n [(2f_d) + f_H] + (f_{id} + f_t) \right] * hop + (2(f_{id} + 2f_H + gf_d + f_t)) \right] \tag{11}$$

Range query schemes for unsorted sensor collected data

Most existing range query processing approaches assume that sensor collected data is sorted[6], [7], [14], [15], [18], [21] or at least uniformly distributed[3] which is not always the case in the real world. Whereas sorted data makes it easy to carry out range queries, sorting data is resource consuming which most schemes ignore. Most sorting algorithms are on average either $O(N^2)$ [9], [10] or $O(n \log n)$ [22] so it is a huge task if sensors have to sort huge amount of data, yet their processing power is limited. It's therefore indispensable to be able to perform privacy-persevering and integrity verifiable range queries on unsorted data. We propose a probabilistic verification Neighborhood chaining approach that can effectively verify range query results.

Probabilistic Verification Neighborhood Chaining (PvNC)

Most existing range query processing approaches assume that the data is sorted[6], [7], [14], [15], [18], [21] or at least uniformly distributed[3] which is not always the case in the real world. Whereas sorted data makes it easy to carry out range queries, sorting data is resource consuming which most schemes ignore. Most sorting algorithms are on average either $O(N^2)$ [9], [10] or $O(n \log n)$ [22] so it is a huge task if sensors have to sort huge amount of data, yet their processing power is limited. It's therefore indispensable to be able to perform privacy-persevering and integrity verifiable range queries on unsorted data. We propose a scheme that we call the Probabilistic Neighborhood chaining (PvNC).

So query communication cost is the summation of the query submission by the sink to \mathcal{M} and results transmission from \mathcal{M} to the sink

$$C_{query} = C_{sink \rightarrow \mathcal{M}} + C_{\mathcal{M} \rightarrow sink} = f_{id} + 4f_H + f_t + f_{id} + 2gf_d + f_t = 2(f_{id} + 2f_H + gf_d + f_t) \tag{10}$$

Total communication cost for AVC protocol is therefore given by

In PvNC, each element is linked with at most ℓ other adjacent elements where $\frac{\ell}{2}$ elements are on either side of d_i . Verification depends on the returned data elements and their association with the dropped elements if any. If any dropped element from the range query appears in any of the PvNC of the returned query results elements, then the integrity of the range query can be verified with high probability. However, if the storage node drops all the data, this scheme can detect that with high precision. The following situations are considered when constructing the PvNC for every element d_i during the time slot t .

- 1) If ℓ is even and d_i has $\left| \ell_{d_{i,left}} \right| \geq \frac{\ell}{2}$ and $\left| \ell_{d_{i,right}} \right| \geq \frac{\ell}{2}$, then d_i is chained with $\frac{\ell}{2}$ neighboring elements on each side.
- 2) If ℓ is odd, and $\left(\left| \ell_{d_{i,left}} \right| \geq \frac{\ell}{2} \right)$ and $\left(\left| \ell_{d_{i,right}} \right| \geq \frac{\ell}{2} \right)$, then $\left| \ell_{d_{i,left}} \right| = \frac{\ell}{2}$ and $\left| \ell_{d_{i,right}} \right| = \left(\frac{\ell}{2} + 1 \right)$.
- 3) If any side of d_i has less elements than $\frac{\ell}{2}$, and the other has more or equal to $\frac{\ell}{2}$, all the elements on the d_i 's side with less elements are chained with d_i and the remainder are chained from the side with more elements.
- 4) If any side of d_i has no elements at all, then all the ℓ elements of PvNC are chained on the other side with elements. This, in most cases happens with the first (d_1) and the last (d_n) elements in the set of data collected by sensor s_i .
- 5) In general, our PvNC is defined as follows:

$$\text{PvNC} = \left\{ \left(d_i \parallel \langle \ell_{d_i, \text{left}} \rangle \circ \langle \ell_{d_i, \text{right}} \rangle \right) \right\} \quad (12)$$

Where \parallel is a concatenation operator and \circ can be any 'separator' symbol for identifying the start and end of individual elements in PvNC after decryption.

$$\text{PvNC} = \{ (6 \parallel \langle \rangle \circ \langle 3 \circ 10 \circ 5 \circ 12 \rangle), (3 \parallel \langle 6 \rangle \circ \langle 10 \circ 5 \circ 12 \rangle), (10 \parallel \langle 6 \circ 3 \rangle \circ \langle 5 \circ 12 \rangle), \dots, (4 \parallel \langle 5 \circ 12 \circ 8 \rangle \circ \langle 9 \rangle), (9 \parallel \langle 5 \circ 12 \circ 8 \circ 4 \rangle \circ \langle \rangle) \}$$

Where $\langle \rangle$ means that either $\langle \ell_{d_i, \text{left}} \rangle = \emptyset$ or $\langle \ell_{d_i, \text{right}} \rangle = \emptyset$.

PvNC Submission Protocol

Like all other schemes suggested in this paper, PvNC makes use of 0-1 encoding scheme [20] in enabling search over encrypted data. The only difference with the above schemes is in the construction of the PvNC. Let sensor s_i data collected during time slot t be given by $D_i = \{d_1, d_2, \dots, d_{n-1}, d_n\}$, where the data is assumed to be in any form, sorted or unsorted.

s_i first builds the PvNC of each and every data item d_i as defined in equation (12) and encrypts it using key k_{it} that is only known by the s_i and the sink. In addition, the sensor builds the corresponding 0-1 encoding values of each element in D_i , and computes enumerated coded numbers (ECN) as defined in equation (3). Let $\text{PvNC}_{k_{it}} = \left\{ \left(d_i \parallel \langle \ell_{d_i, \text{left}} \rangle \circ \langle \ell_{d_i, \text{right}} \rangle \right)_{k_{it}} \right\}$ be the encrypted PvNC of d_i during time slot t . The sensor node s_i submits the following message to the storage node.

$$s_i \rightarrow \mathcal{M}: i, t, \{ \text{PvNC}_{k_{it}}, \text{ECN}_i \}$$

When there is no data collected during time t , s_i sends the system data item that will be used by the sink in case the range query result set is empty. As already discussed above, the d_{\min} and d_{\max} are supposed to be either far below or far above the expected domain minimum or maximum such that at no particular time any node collects any $d_i = d_{\min}$ or $d_i = d_{\max}$

$$s_i \rightarrow \mathcal{M}: i, t, \{ (d_{\min})_{k_{it}}, (d_{\max})_{k_{it}}, \text{ECN}_{\min}, \text{ECN}_{\max} \}$$

The PvNC Range Query Submission and Processing Protocols

Query submission protocol is concerned with how the sink submits its query to the storage node. Given a range query $[a, b]$, the sink computes the 0-1 coding of a and b and constructs a query as shown below

Example: We now explain our PvNC with an example. Let $D_i = \{6, 3, 10, 5, 12, 8, 4, 9\}$ and $\ell = 4$. This means that each element should be chained with 4 adjacent elements.

$$\text{sink} \rightarrow \mathcal{M}: i, t, \{ \text{ECN}(a), \text{ECN}(b) \}$$

Upon receiving the range query from the sink, \mathcal{M} looks up all the encrypted PvNC data that satisfy the range query as by comparing every $\text{ECN}(d_i)$ with $[\text{ECN}(a), \text{ECN}(b)]$ where $d_i \in [a, b]$ and within the collected data being queried upon. d_i qualifies to be in the range query results if and only if $(\text{ECN}^1(a) \cap \text{ECN}^0(d_i) \neq \emptyset) \wedge (\text{ECN}^1(d_i) \cap \text{ECN}^0(b) \neq \emptyset)$ as defined in section 4.1

\mathcal{M} sends the following response to the sink.

$$\mathcal{M} \rightarrow \text{sink}: \{ i, t, Q_{\text{result}} \}$$

In this case, $Q_{\text{result}} = \{ \text{PvNC}_{k_{it}} \}$ i.e, a set of PvNC that satisfy the range query.

Upon receiving the range Query results, the sink decrypts the received PvNC data items and tries to verify the authenticity of the results. The results can only be accepted if the following conditions apply.

1) Data can be decrypted using known keys. If the received data items cannot be decrypted using the keys that the sensor shared with the sink, then data can be assumed to be invalid.

2) The received PvNC data satisfies the query i.e all the received data is within the range $[a, b]$.

3) After verifying the received PvNC data items, if any data item that satisfies the range query was eliminated from the range query results, yet it appears somewhere within any of the received data items' Probabilistic Verification Neighborhood chaining, then the range query results are incomplete and should be rejected.

4) If the query result is empty, then the storage node should send the secret minimum data element that was sent by the sensor during time t when it did not collect any data items. This minimum only exists if and only if the sensor did not collect any data during time interval t .

Communication cost for PvNC

We consider communication cost incurred by both the sensor nodes in transferring data to the storage nodes and the communication cost incurred

when the storage node sends range Query results to the sink. Let $C_{pvnc \rightarrow \mathcal{M}}$ be the communication cost for each sensor data submission to \mathcal{M} . We consider a cell with n sensors,

$$C_{pvnc \rightarrow \mathcal{M}} = \sum_{i=1}^Q \left[\left[\sum_{j=1}^n [(f_d + \ell f_d + 2f_H + f_{sep})] \right] + (f_{id} + f_t) \right] * hop \tag{13}$$

Where ℓ is the number of data elements to be chained with every d_i and f_{sep} is the size of the separator symbol.

Communication cost for range query in PvNC

The cost of sensing range query to the storage node is the same for all our schemes and is given by equation (8). Assuming \mathcal{M} returns g PvNC items to the sink, the communication cost involved is given by

Communication Cost for Sensor Node In PvNC

Still referring to the same definitions in table 1, the communication cost for PvNC is given by

$$C_{pvnc \rightarrow sink} = f_{id} + \ell g(f_d + f_{sep}) + f_t \tag{14}$$

Where f_{sep} is the size of the separator symbol as described in section 5.1.1. Combining equation (8) and (14) gives

$$C_{QPvnc} = 2(f_{id} + f_t + 2f_H) + \ell g(f_d + f_{sep}) \tag{15}$$

Combining equation (13) and (15) gives the total communication cost incurred by the PvNC scheme as

$$C_{pvnc_Total} = \sum_{i=1}^Q \left[\left[\sum_{j=1}^n [(f_d + \ell f_d + f_H + f_{sep})] \right] + (f_{id} + f_t) \right] * hop + 2(f_{id} + f_t + 2f_H) + \ell g(f_d + f_{sep}) \tag{16}$$

Probability of Range Query Results Integrity verification for PvNC protocol

Since all the expected range query results data items are not next to one another, determining whether \mathcal{M} dropped some elements that satisfy the range query is challenging in PvNC. PvNC can only verify that some data was eliminated from the range query results if and only if the deleted item appears in any of the received PvNC data structures. However, since the adversary does not know in advance what each data element is concatenated with, determining what to drop such that he is not detected is also a challenging task to him. This means that if the adversary decides to drop some elements from the range query results, he must drop them with the same probability.

The success of PvNC depends on a number of different factors: 1) The number of data elements linked to one another (ℓ), 2) the number of data elements returned in the range query and 3) The number of data elements dropped by the adversary. The higher the value ℓ , the more likely \mathcal{M} will be detected for dropping some of the data. Similarly, the higher the number of elements returned in the range query, the more likely the relationship between more elements collected by s_i and hence easier to detect dropped elements by examining the received elements. On a similar note, the higher the number of elements

dropped by adversary from the range query results, the easier it is to detect him. Every element dropped \mathcal{M} increases the chances of \mathcal{M} being detected by ℓ times. Because the sink does not know in advance how many data elements each range query will return, his main objective is not to detect how many data elements where dropped, but to detect if any data element was dropped which suggests malicious action from \mathcal{M} . For example, if \mathcal{M} drops 100 data elements from the range query results and the sink is able to detect one of the dropped data, that is enough to suspect \mathcal{M} of being malicious. The fewer the data elements dropped, the better the results.

If n is the number of data elements in the given time slot and ℓ is the number of data elements each element is linked with, for any data element dropped by \mathcal{M} , there are ℓ possibilities of detecting the dropped data from the range query results. Let w the number of data elements dropped by \mathcal{M} from the range query results. Then, there are $\ell * w$ of detecting any dropped data from the collected n elements. This implies that the large the number of data elements dropped by the adversary, the easier it becomes to be detected. In addition, the fewer the elements dropped, the lower the negative effect on the overall results. Our aim is to reduce the number of elements that can be dropped from the range query

results as much as possible. From the above analysis, when \mathcal{M} drops more number of data elements, it risks being detected yet it would like to remain undetected as much as possible.

To verify this, we ran two experiments where we varied the value ℓ in one and value of w in the other. In the first case, for each value of ℓ , we define range queries and let \mathcal{M} randomly drop an arbitrary number of data elements from the results. Then we let the sink determine the number of dropped elements. In the second experiment, we let \mathcal{M} drop a known number of elements from the range query results (but the sink does not know) and let the sink determine how many

dropped elements it can identify from the range query results. The goal here is to estimate how many data elements were dropped by \mathcal{M} from the received range query results. Results, as shown in **Figure 2.**, reveal that the higher the value of ℓ , the more dropped elements can be identified by the sink irrespective of the size of n . Similarly, Figure 3 also reveals that, the more the number of data elements deleted the high the probability of detecting dropped elements. Therefore, in order for \mathcal{M} to escape detection, it has to drop as minimum data as possible if it must and that is our goal; to minimise data loss as much as possible.

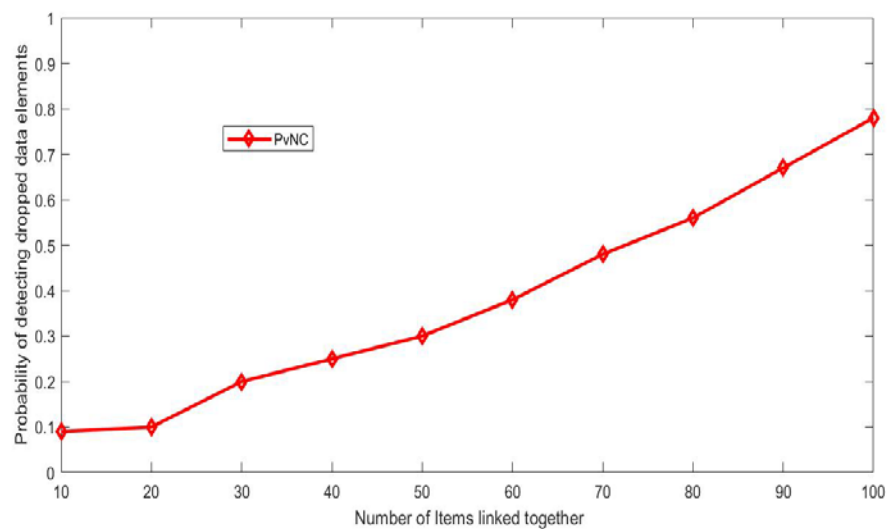


Figure 2 – Probability of detecting if any data was dropped depending on the number of items linked together in PvNC Scheme

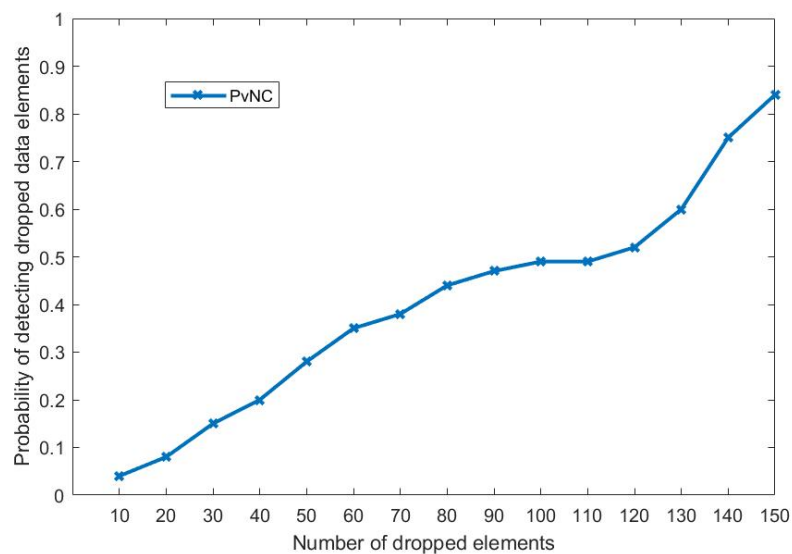


Figure 3 – Probability of detecting if any data was deleted from range query results with increase in the number of items dropped by the adversary

Experimental Results

Experimental setup: We simulated our schemes using OMNET++5.1 simulator [23] on an Intel Core i7 CPU machine, 8gb of RAM on a large real dataset from Intel Lab [24] and the results were further analyzed and evaluated using MATLAB. But unlike in [14], we only conducted experiments on a single dimension data set, that is temperature. We compare the communication cost of our AVC scheme with CSRQ [7] and PRQ [15] under a non-collusion environment. We chose these two schemes for comparison because of the close relationship between them and AVC our model. Secondly, several other schemes like SafeQ [14], encoding scheme[6] were extensively compared with CSRQ in [7] on the same dataset [24] which was also used in this study. The performance comparisons of our AVC scheme with CSRQ indirectly reveals the performance of our model in comparison with those considered in [7]. The network is set to 200m X 200m containing four

cells with approximately equal number of sensors and a storage node at the center of each cell. We vary the number of sensors between 50 to 200. The average transmission distance of each sensor to the storage node is 30m. All other parameters used in the experimental setup are shown in Table 2.

Measurement Metrics: In our experimental evaluations, we considered the efficiency of the schemes as well as range query accuracy. Efficiency, in this case, refers to both node and range query power consumption of the scheme in consideration. We define accuracy as the number of false positives in the range query results. False positive is defined as the number of unsatisfactory data in the range query result. The lesser the false positives, the better the accuracy.

Besides comparing our AVC model with other related models, we evaluate the communication cost as well as the accuracy of our model for unsorted data. Experimental results drastic reduction in communication cost for the PvNC as shown in Figure 4.

Table 2 – Experiment Parameters

Parameter	Value	Parameter	Value
Network area(m^2)	200x200	Maximum number of sensors	400
Size of data item(f_d)	32 bits	Size of sensor id(f_{id})	32 bits
Size of the time slot t(f_t)	32 bits	Maximum Number data items per time slot	2000000
Number of elements(ℓ) chained with d_i in PvNC	4-100	HMAC function	128 bits

Discussion of results

In Figure 4, we compare the communication cost of our models with two other existing models that are closely related with AVC model, that is the CSRQ [7] and PRQ [15]. As can be observed from the figures, CSRQ and PRQ outperform our model for unsorted data- PvNC in terms of communication cost. However, AVC favorably compares with these existing models in terms of communication cost much is CSRQ is still slightly better. However, CSRQ suffers from more false positives when compared to AVC as illustrated in Figure 5.

Figure 5 displays the impact of number of data as well as the number of range query results on the false positives generated by different schemes. Note that the number of sensor nodes is directly proportional to the number of data elements. AVC, PvNC and PRQ schemes have less percentage of false positive because they make use of 0-1 encoding as well as BF codes (PRQ) which can securely identify encrypted data without generating many false positives. CSRQ

uses data structure called Encrypted constraint chain in which numbers stored in ascending order are partitioned into several parts with a given parameter, encrypted separately and can be brought together by concatenating the different parts together. Because different Encrypted Constraint chains have some data in common, false positives are possible.

Figure 6 reveals the impact of number of data elements on the power consumption. As stated in [17], time spent processing data is directly proportional to the power consumption of the sensors. In our experiments, we varied the number of data items and recorded the corresponding time each scheme takes to submit its data to the storage node. We used the same DES encryption algorithm preserving privacy and the same HMAC-MD5[25] algorithm for all the schemes that require the use of hash functions. We experimented using three sorting algorithms for schemes that require data to be sorted that is bubble sort, insertion sort and quick sort. Experimental results show that there is a big impact on power consumption of the sensor nodes depending

on the sorting algorithm used. For example, it took AVC, PRQ and CSRQ approximately 67.212 seconds to submit 200,000 data elements using bubble sort, while it took 22.571 and 2.52 using insertion and quick sort respectively. Because using quick sort produced the shortest submission time, our subsequent discussions ignore other sorting algorithms that produced more submission time. The network owner is expected to choose the best sorting algorithm that suits the resource consumption need of a particular sensor. We note that the more the number of data elements to be sorted, the higher the submission time hence high power consumption. Our results still reveal a significant difference in submission time between schemes that require data to be in sorted form and our schemes for unsorted data. For example, it took averagely 185.813 seconds for AVC, PRQ and CSRQ schemes to submit 1.5

million data items to the storage node, while it took 85.012 seconds for PvNC to submit the same number of data elements. The explanation to this observation is that AVC, PRQ and CSRQ use most of the time sorting the data, before starting the process of encrypting and subsequent submission to the storage node. We believe these figures could go down if machines with higher resources are used. However, sensors have limited resources in terms of storage and processing power yet a lot of data can be collected in a very short period by the sensor. Our schemes for unsorted data out-perform the schemes that assume sorted data especially for a large number of data but still guarantee privacy of the data. In conclusion, experimental results reveal that PvNC greatly reduces the power consumption of the sensor, especially for big number of data elements compared to the existing schemes.

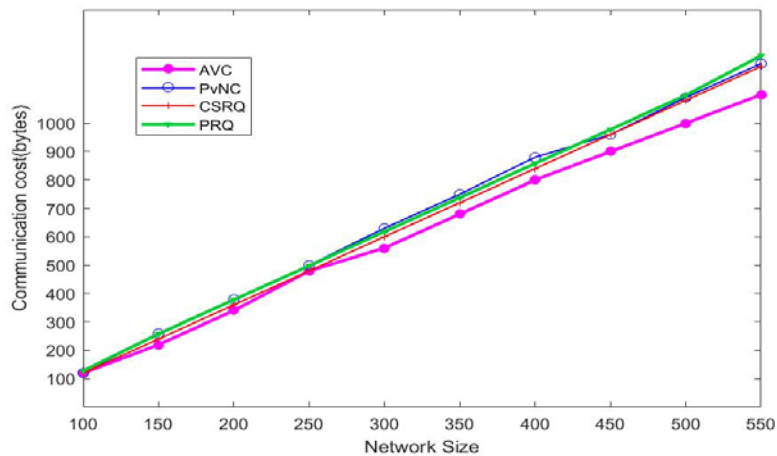
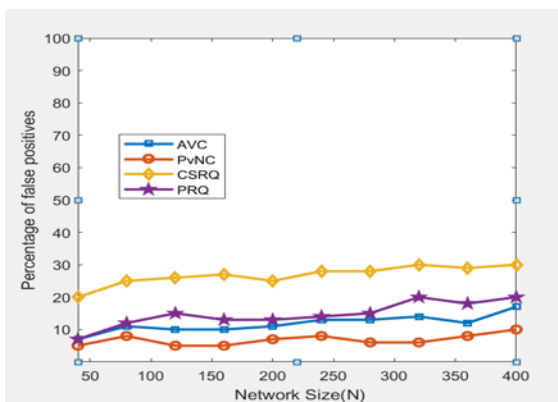
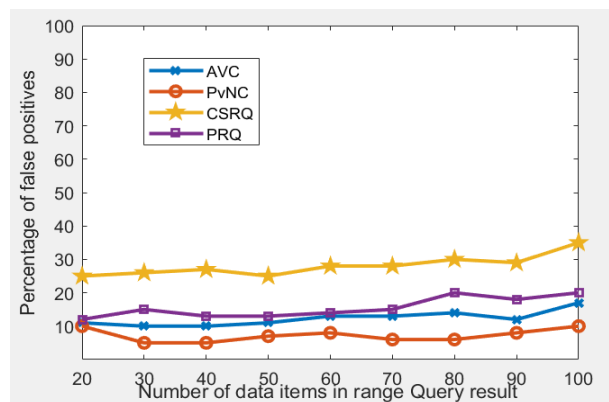


Figure 4 – Impact of Network size on Communication cost- Comparison between our models and selected previous models



(a) Impact of sensor nodes on false positives



(b) Impact number of query results items on false positives

Figure 5 – Impact of number of data elements and network size on False Positives of different schemes

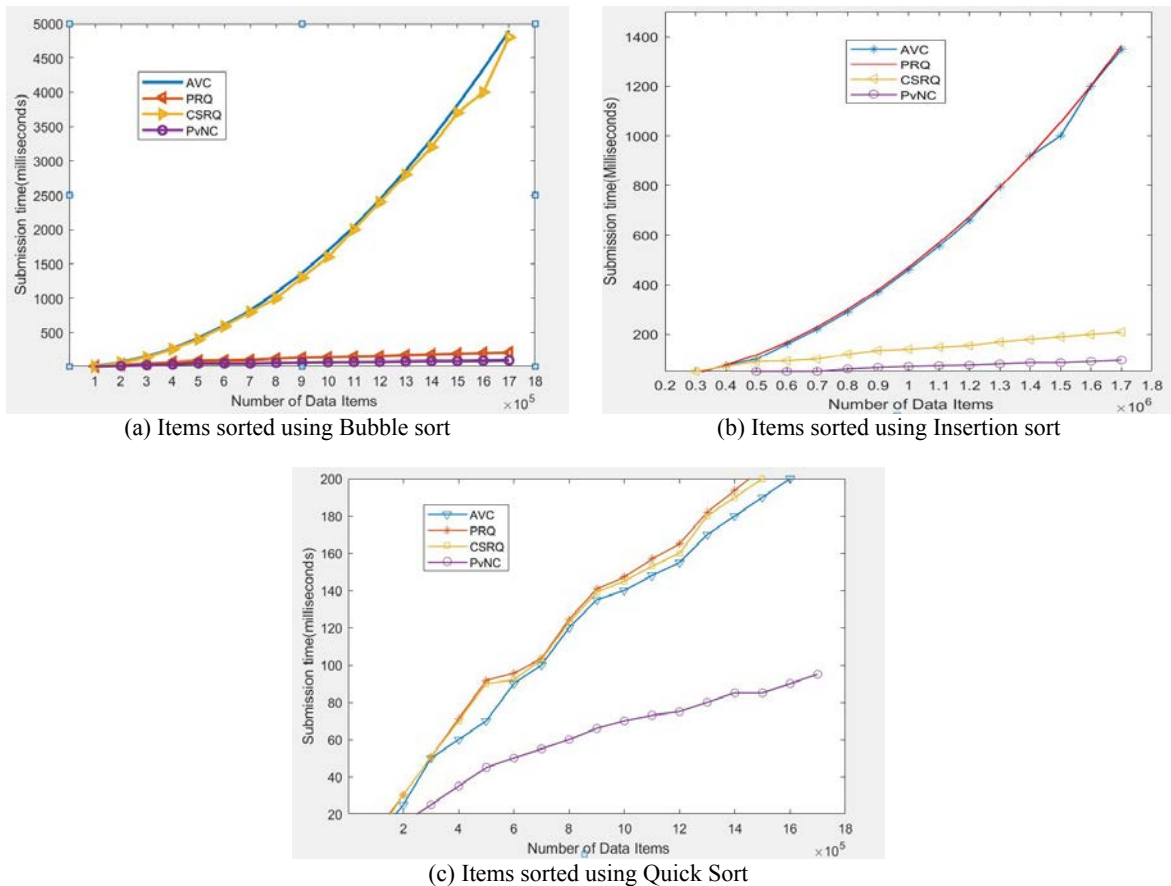


Figure 6 – Impact of number of data Items on Sensor Nodes Submission time of different schemes

Conclusion

In this paper, we make three contributions. First, we propose AVC protocol, which assumes that the sensor data is in sorted order just like the existing schemes. We further propose a scheme for unsorted data, the PvNC protocol. The results of our experiments show that PvNC has a better performance in terms of power consumption than existing range query protocols. We observe that having to sort data elements before being submitted to the storage node increases power consumption of the sensor nodes yet sensors generally have low processing power. While sorting data simplifies the range query processing at the storage node, the high power consumption rate by sensor nodes cannot be ignored. Our scheme can easily be extended to Numerical multidimensional data by computing the 0-encoding and 1-encoding of each attribute separately and then can be concatenated to represent a tuple.

Funding: This work was supported by Hunan province science and technology project fund (2018TP1036).

Data Availability: We used publically and freely available data set from Intel Lab collected from 54 sensors deployed in the Intel Berkeley Research lab between February 28th and April 5th, 2004 that can be accessed at <http://db.csail.mit.edu/labdata/labdata.html>.

References

- 1 Kottapalli VA, Kiremidjian AS, Lynch JP, et al. Two-tiered wireless sensor network architecture for structural health monitoring. *Proc SPIE*. 5057 (2003): 8-19. doi:10.1117/12.482717.
- 2 RISE: More powerful, Energy efficient, Gigabyte scale storage high performance sensors. <http://www.cs.ucr.edu/~rise/>.
- 3 Li R, Liu AX, Xiao S, Xu H, Bruhadeshwar B, Wang AL. Privacy and Integrity Preserving Top-

- k Query Processing for Two-Tiered Sensor Networks. *IEEE/ACM Trans Netw.* 25(4) 2017:2334-2346. doi:10.1109/TNET.2017.2693364.
- 4 Dai HUA, Wang MIN, Yi XUN, Yang G, Bao J. Secure MAX / MIN Queries in Two-Tiered Wireless Sensor Networks. *IEEE Access.* 2017;5. doi:10.1109/ACCESS.2017.2731819.
- 5 Bu J, Yin M, He D, Xia F, Chen C. SEF: A secure, efficient, and flexible range query scheme in two-tiered sensor networks. *Int J Distrib Sens Networks.* (2011). doi:10.1155/2011/126407.
- 6 Sheng B, Li Q. Verifiable privacy-preserving sensor network storage for range query. *IEEE Trans Mob Comput.* 10(9) (2011):1312-1326. doi:10.1109/TMC.2010.236.
- 7 Dai H, Ye Q, Yang G, Xu J, He R. CSRQ: Communication-efficient secure range queries in two-tiered sensor networks. *Sensors (Switzerland).* 2016;16(2). doi:10.3390/s16020259.
- 8 Chen F, Liu AX. SafeQ: Secure and efficient query processing in sensor networks. *Ieee Infocom 2010.* (2010). doi:10.1109/INFCOM.2010.5462094.
- 9 Yang Y, Yu P, Gan Y. Experimental study on the five sort algorithms. In: *2011 Second International Conference on Mechanic Automation and Control Engineering, Hohhot.* (2011):1314-1317. doi: 10.1109/MACE.2011.5987184.
- 10 Edjlal R, Edjlal A, Moradi T. A sort implementation comparing with Bubble sort and Selection sort. In: *2011 3rd International Conference on Computer Research and Development.* Vol 4. (2011): 380-381. doi:10.1109/ICCRD.2011.5763927.
- 11 Hacigümüş H, Iyer B, Li C, Mehrotra S. Executing SQL over encrypted data in the database-service-provider model. *Proc 2002 ACM SIGMOD Int Conf Manag data – SIGMOD '02.* 7 (2002) :216. doi:10.1145/564716.564717.
- 12 Shi J, Zhang R, Zhang Y. A spatiotemporal approach for secure range queries in tiered sensor networks. *IEEE Trans Wirel Commun.* 10(1) (2011): 264-273. doi:10.1109/TWC.2010.102210.100548.
- 13 Zeng J, Dong L, Wu Y, Chen H, Li C, Wang S. Privacy-preserving and Multi-dimensional Range Query in Two-tiered Wireless Sensor Networks. In: *GLOBECOM 2017 – 2017 IEEE Global Communications Conference.* (2017): 1-7. doi:10.1109/GLOCOM.2017.8254968.
- 14 Chen F, Liu AX. Privacy- and Integrity-Preserving Range Queries in Sensor Networks. *IEEE/ACM Trans Netw.* 20(6) (2012): 1774-1787. doi:10.1109/TNET.2012.2188540.
- 15 Zhang X, Dong L, Peng H, Chen H, Zhao S, Li C. Collusion-Aware Privacy-Preserving Range Query in Tiered Wireless Sensor Networks. *Sensors (Switzerland).* (2014): 23905-23932. doi:10.3390/s141223905.
- 16 He R, Dai H, Yang G, Wang T, Bao J. An Efficient Top-k Query Processing with Result Integrity Verification in Two-Tiered Wireless Sensor Networks. *Math Probl Eng.* 2015; (2015). doi:10.1155/2015/538482.
- 17 Li R, Liu AX, Wang AL, Bruhadeshwar B. Fast and Scalable Range Query Processing with Strong Privacy Protection for Cloud Computing. *IEEE/ACM Trans Netw.* 24(4) (2016): 2305-2318. doi:10.1109/TNET.2015.2457493.
- 18 Wen M, Lu R, Zhang K, Lei J, Liang X, Shen X. PaRQ: A privacy-preserving range query scheme over encrypted metering data for smart grid. *IEEE Trans Emerg Top Comput.* 1(1) (2013): 178-191. doi:10.1109/TETC.2013.2273889.
- 19 Tsou YT, Lu CS, Kuo SY. Privacy- and integrity-preserving range query in wireless sensor networks. *GLOBECOM – IEEE Glob Telecommun Conf.* (2012): 328-334. doi:10.1109/GLOCOM.2012.6503134.
- 20 Lin H-Y, Tzeng W. An Efficient Solution to the Millionaires' Problem Based on Homomorphic Encryption. *ACNS 2005, Vol 3531 Lect.* (2005): 456–466. doi:10.1007/11496137_31.
- 21 Shen Y, Yang W, Li L, Huang L. Achieving fully privacy-preserving private range queries over outsourced cloud data. *Pervasive Mob Comput.* 39 (2017): 36-51. doi:10.1016/j.pmcj.2017.04.008.
- 22 Xiang W, Edjlal R, Edjlal A, Moradi T. Analysis of the Time Complexity of Quick Sort Algorithm. In: *2011 International Conference on Information Management, Innovation Management and Industrial Engineering.* Vol 1.; (2011): 408-410. doi:10.1109/ICIII.2011.104.
- 23 Varga A. Using the OMNeT ++ Discrete Event Simulation System in Education. *IEEE Trans Educ.* 42(4) (1999): 11.
- 24 Bodik P, Hong W, Guestrin C, Madden S, Paskin M, Thibaux R. Intel Lab Data. MIT. <http://db.csail.mit.edu/labdata/labdata.html>. Published 2004. Accessed June 20, (2018).
- 25 Krawczyk H, Bellare M, Canetti R. HMAC: Keyed-hashing for message authentication. In: *RFC 2104;* (1997).