國立交通大學

資訊科學與工程研究所

碩士論文

# 無線感測網路中基於資料聚集方法下的安全比較系統

# A Secure Comparison Protocol with Data Aggregation for Wireless Sensor Networks

研 究 生：陳宏達

指導教授：曾文貴 教授

中 華 民 國 九 十 七 年 六 月

# 無線感測網路中基於資料聚集方法下的安全比較系統

# A Secure Comparison Protocol with Data Aggregation for Wireless Sensor Networks

研 究 生: 陳宏達　　　　Student: Hung-Ta Chen

指導教授: 曾文貴　　　　Advisor: Wen-Guey Tzeng

國立交通大學

資訊科學與工程研究所

碩士論文

A Thesis
Submitted to Institute of Computer Science and Engineering
College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
For the Degree of
Master
In

Computer Science

June 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年六月

## 中文摘要

無線感測網路在最近幾年快速的發展，廣泛的應用在民生和軍事上，因此安全性的議題也就漸漸被重視。由於感測器是擁有有限的電力，計算能力以及儲存空間，因此要在無線感測網路上建立一套有效率且安全的協定並不容易。關於改善效率，在無線感測網路上存在一個解決方法，稱之為—Data Aggregation(資料聚集)，此方法可以有效的降低資源的消耗以及通訊頻寬。不過由於資料聚集的方式必須是要能在中間的節點 (Aggregator)負責處理資料的整合和計算，因此如何在資料聚集的方法下提供 end-to-end 的資料保密性就變得是一項挑戰。在近幾年間，有一些研究在解決上述的問題，不過大多數都是集中在解決某一種資料聚集函數像是”加總”。

然而，在感測網路的應用當中也有需要做中位數或是尋找最大最小值等需要做比較運算的資料聚集函數。因此在這篇論文中，我們利用了輕量化的加密演算法且能提供在密文上做資料計算的方法 Privacy Homomorphism 來達成我們的目標。此外，透過運用在密碼上解決比大小問題的機制我們提出了一套在無線感測網路架構上安全比較的方法，此方法可以做到有效的降低資源的消耗且能在密文上做比較的運算來找出最大最小值。

關鍵字: 資料聚集，機密性，感測網路，安全計算

## Abstract

In recent years, wireless sensor networks (WSNs) continue to grow rapidly and have widely use in both military and civilian. Therefore, security issues of WSNs are more important. Since the sensor nodes have limited power, computation, and storage, it is not easy to establish efficient and secure protocol for WSNs. In terms of efficiency, a countermeasure -- *Data Aggregation* has been proposed for reducing the resource consumption and communication bandwidth. Since data aggregation needs to compute and aggregate data at the intermediate nodes (Aggregators), how to provide end-to-end privacy with data aggregation in WSNs becomes a challenge. In recent years, some research is interesting in above mentioned secure aggregation for WSNs, but most of them focus on a certain aggregation function like "SUM".

However, a large class of sensor network applications such as median computation or finding maximum/minimum, rely on comparison operations. In this paper, we use a light-weight encryption scheme called *Privacy Homomorphism* that supports operations over ciphertext for our purpose. Additionally, we propose a secure comparison protocol that achieves the energy benefits and support secure comparison operations over the encrypted values for WSNs by some techniques used to solve "Grater than" problem in cryptographic.

**Key words**: *data aggregation, confidentiality, sensor network, Secure Computation*

# Content of Table

# List of Tables

# List of Figures

# 1. Introduction

A wireless sensor network (WSN) is composed of many small sensor nodes deployed to sense the environment. The main element of constructing a sensor network is sensor nodes. A sensor is a programmable microcontroller with sensing component, which is a low-cost device used to sense and collect environment information, such as temperature, humidity, light, magnetic, acceleration, acoustic, etc[1]. There is one base station (or sink) which is used to find the summarized statistics of the whole network. The base station diffuses a specific task to whole network and the sensors collect raw data for a given task and then report to the base station. Wireless sensor networks continue to grow rapidly with cheaper price characteristics of sensor nodes. Sensor nodes are low cost solutions to a variety of real-world applications. WSNs have widely use in both military and civilian. The applications include real-time traffic monitoring, military surveillance, tracking at critical facilities, monitoring as animal habitats, detecting blaze in forest, etc. Sensor networks change the way people interact with the environment [1][2].

Since sensor nodes have limited power, computation, and storage, it is not easy to establish efficient and secure protocol for WSN. In terms of efficiency, most of sensor nodes consume energy during computation and transmission of data packets. Therefore, we need to reduce the amount of raw data sent by processing in-network and a sensor network. We refer to in-network processing as data aggregation. The idea of data aggregation is to combine several readings at intermediate nodes for saving bandwidth and computation. We describe details of data aggregation in the following section.

Another important issue about a wireless sensor network is security, especially in military usage. Sensor nodes are deployed in the hostile and unsecure environment.

Due to low cost and resource constraint, a sensor is not suitable to equip with a tamper-resistant device. Also, the unreliable communication channel of sensor networks make defense become more hard, so an attacker can take control of several nodes and private data can be stolen. Additionally, collected data can be modified or erased and then direct the base station to agree on the false result. For these reasons, a lot of research has been proposed to solve various aspects of sensor network security in recent years, like data integrity/authentication, broadcast authentication, key management, location verification and location privacy, secure routing and forwarding. Furthermore, because of nodes compromised is a serious threat of sensor networks, some research focuses on detection of compromise and revocation.

Data privacy is a basic security requirement for wireless sensor networks, especially in a battleground where data is sensitive. Although some research on protecting the sensitive data from eavesdropping have existed, most of them put emphasis on special kind of aggregation functions, such as SUM, Variance, and Average. With comparison function like MAX/MIN, they don’t refer to such aggregation function. In addition, it represents major obstacles to the implementation of traditional public key based cryptosystems in sensor nodes because of energy consumption.

In this paper, we focus on the *data confidentiality* and provide secure comparison to solving the problem of finding maximum/minimum with in-network aggregation. We use a light-weight cryptographic encryption scheme called *privacy homomorphism* proposed by Domingo-Ferrer [20] instead of public key based cryptosystems. By using privacy homomorphism, we aggregate data in an encrypted form at the aggregator and provide end-to-end privacy. Additionally, we first utilize some cryptographic techniques [16] to process secure comparison in encrypted data for wireless sensor networks. Our protocol limits the adversary’s ability to obtain exact information about normal sensor nodes and robust against known-plaintext attack.

Since sensor nodes can be compromised by attacker, attacks such as forging or modifying messages, can be carried out easily. Hence, other security requirements like data integrity/authentication are equally important issues. Those requirements prevent the compromised nodes from misleading the base station to agree on an incorrect value. However, we ignore data integrity and authentication in this paper, and we develop our protocol by some literature about data integrity/authentication in existence.

In summary, the remainder of this paper is organized as follows. Section 2 introduces some related work for sensor network security and notion of data aggregation. Section 3 describes some preliminaries which are related to our protocol. Section 4 explains the protocol model and background, including problem definition, attacker model, and some assumption for our protocol. The main protocol description and security analysis are presented in Section 5 and Section 6, respectively. In Section 7 we discuss the overhead of our protocol. Finally, we give a conclusion and future work in Section 8.

# 2. Related Work

In the beginning, many literature have been proposed for minimizing energy consumption in WSN [3][17][18]. They present a technique called data aggregation, which can decrease the energy consumption efficiently. But they assume all nodes in the network are honest, and none of them integrate the security threats with in-network aggregation.

In recent years, a lot of research has been proposed to solve the problem about sensor network security with data aggregation. In order to defeat an active adversary whose goal is to tamper or discard messages such that the base station obtains a wrong result, some research discusses data integrity and data authentication for wireless sensor networks [4][5][10][13]. Those works have contribution that the base station accepts the aggregation result with high probability if the aggregated result is on a desired bound. In other words, the base station rejects wrong results (out of desired bound) and detects the compromised nodes.

Due to lots of applications on collecting sensitive measurement, some research focuses on data confidentiality [6][7][8][11]. In [6] Girao et al. introduces a concept of Conceal Data Aggregation, which is the first work in providing end-to-end privacy for wireless sensor networks. They provide a solution for processing encrypted data at the intermediate nodes (aggregators) by using *privacy homomorphism*. This work reforms the disadvantage of hop-by-hop encryption schemes which we introduce in the next section.

In [7], Castelluccia et al. suggests another approach to aggregate encrypted data for the SUM aggregation. In contrast to [6], they propose a simple and provably secure additively homomorphic scheme that process encrypted data efficiently. The homomorphic encryption scheme used in their architecture is simple and provable secure. This scheme is illustrated as follows.

**Additively Homomorphic Encryption Scheme proposed by Castelluccia et al.**

*System Setup*:

- $M$ is a large integer. $m$ denote a plaintext and $m \in [0,...,M-1]$.
- Each sensor nodes share a unique pair-wise key $k$ with base station. Let $k \in_R [0,...,M-1]$
- Let *Enc*( ) denotes encryption function and *Dec*( ) represents decryption function.

*Encryption*:

Compute $c = Enc(m, k, M) = m + k \pmod M$

*Decryption*:

Compute $m = \mathrm{Dec}(c, k, M) = c - k \pmod M$

*Addition Homomorphism*:

1. Let $c_1 = Enc(m_1, k_1, M) = m_1 + k_1 \pmod M$ and
   $c_2 = Enc(m_2, k_2, M) = m_2 + k_2 \pmod M$
2. $m_1 + m_2 = \mathrm{Dec}(c_1 + c_2, k_1 + k_2, M) = (c_1 + c_2) - (k_1 + k_2) \pmod M$

Although this scheme is cheaper than [6] in resource consumption, and provides security analysis, but it has some limitation. First, the key length must be as long as plaintext and the key management is also problematic. Second, this method is not suitable for other aggregation function such as MAX/MIN. Acharya et al. [8] shows that the first secure comparison scheme which allows comparison operation performed on ciphertex. This scheme uses another encryption scheme that can preserve the order of plaintext. But in this approach, it is only secure against ciphertext-only attack. In other words, if one of sensor nodes is compromised, the adversary is able to get the plaintext-ciphertext pair. Therefore, the privacy of data is broken over whole networks. In [11], He et al. proposed two efficient privacy-preserving data aggregation protocols called CPDA and SMART, the used technique of both schemes differ from [6][7], CPDA uses algebraic properties of polynomials to compute the aggregate value. In the SMART scheme, each node splits its private value into pieces and sends encrypted partial values to other nodes. Then the other nodes can calculate the aggregate value. Finally, all of partial aggregate values are collected by the base station. These approaches are efficient and energy-saving, but it works for statistical functions such as SUM and AVERAGE. Jadia et al. and Kifayat et al. [9][14], they combine several security requirements such as data confidentiality and authentication mechanism for establishing secure data aggregation protocols.

There are plenty of multiparty secure computation (SMC) techniques used in cryptography, SMC proposes a solution for the problem of processing encrypted data. In [16], Chu et al. presents a fundamental scheme that is useful in construction secure interactive protocols, they propose schemes for "equality", "inequality" and "greater than" predicates. Because of these schemes are not computationally expensive for WSNs, we use the proposed technique to establish secure protocols for performing MAX/MIN functions in wireless sensor networks.

In the following two sections, we discuss data aggregation in WSNs and show how to process encrypted data with data aggregation.

## 2.1 Data Aggregation in Wireless Sensor Networks

Because of the resource and power restriction of a sensor node, data aggregation is used to reduce the data communication cost and energy consumption of sensor networks. Many works have been proposed in recent years [3][17][18]. Before a data aggregation, sensor nodes are formed into a hierarchical cluster-based tree structure. In this tree structure, the base station is the root of a tree, and sensor nodes spilt into several clusters. Within a cluster, one of sensor nodes is elected as the *aggregator*, the remainding nodes become *sensing nodes*. The aggregators are formed into a tree structure. Moreover, they can be elected dynamically to balance the power consumption of all the nodes [20]. However this issue is out of scope of this paper. The tree structure is illustrated in Figure 1. Base on the operation of wireless sensor networks, each node measures sensitive data periodically. When data are taken by individual sensing nodes, they need to be collected and processed to output the result by specific aggregation function, such as MAX/MIN, SUM, AVERAGE, VARIANCE, etc. In order to save the bandwidth and energy of nodes, an approach is to send this collected data to certain special node. More accurately, we refer to some special nodes as the *Aggregators*. Then aggregators exploit some arithmetic operation for data aggregation. Next, aggregators send the partial result to upper layer cluster for next aggregation. Eventually, partial results will aggregate at sink (base station). The aggregator can either be more powerful nodes or regular sensor nodes. In this paper, we assume aggregators are elected randomly from sensor nodes. Hence, the aggregators must require simple arithmetic operations, such as additions or subtractions or multiplications.

Without considering the security, for some statistical measurements like SUM, AVERAGE aggregation functions, a general method is to simply add up values received from its child nodes and then forwards the partial result to base station. For aggregation functions like MAX/MIN, we also can process it by the order of the value. However, the assumption that all sensor nodes are honest is an unrealistic assumption in a wireless sensor network. We will then discuss how to provide security with data aggregation for WSN in next section.

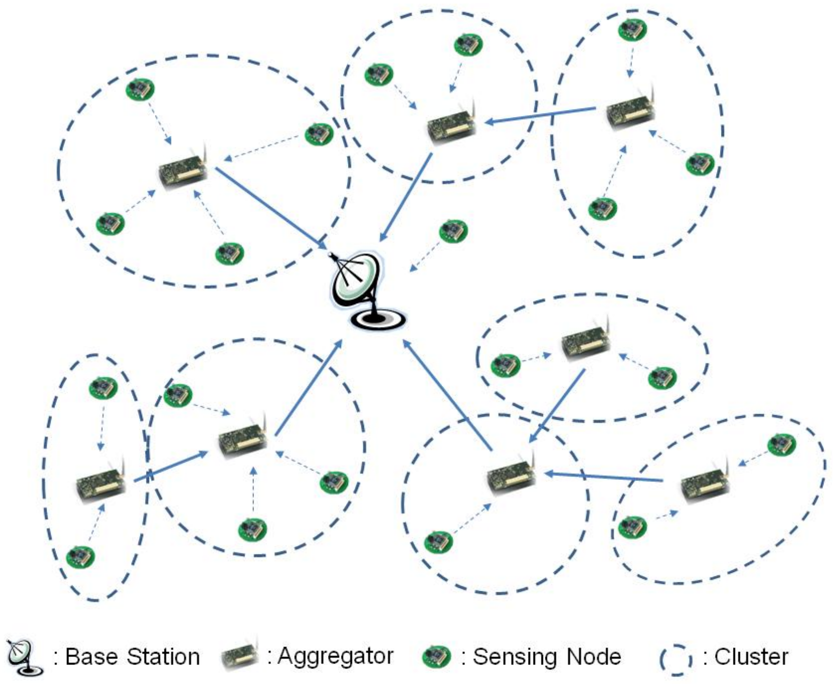


**Figure 1. The cluster-based aggregation tree structure.**

## 2.2 Aggregation for Encrypted Data

As mentioned before, the data aggregation reduces the amount of communication within wireless sensor networks, and lets the procedure run more efficiently. While if data confidentiality is required, efficient data aggregation becomes a challenge. There are some solutions for providing data privacy within network. The standard approach to protect the private information is to encrypt the sensitive data with a secret key that only the legal receiver can decrypt it. In wireless sensor networks, sensor nodes encrypt their private data by using a unique shared secret key with base station and then forward the encrypted value to base station through other nodes. Upon receiving all response messages from sensor nodes, base station decrypts all the ciphertext, and then aggregates them according to specific aggregation function. This kind of solution achieves end-to-end privacy, but has obvious drawback. Since sensor nodes transform packets to base station directly, it steps up traffic within the network enormously.

Another solution is called *hop-by-hop* (HBH) encryption. The general idea of HBH encryption is composed of three phases,

1) The bootstrapping phase: In this phase, to establish secure link between a cluster leader (aggregator) and sensing nodes by using a pair-wise key sharing approach.
2) The data aggregation phase: Within the cluster, children nodes encrypt their readings by shared key with the aggregator *A*, and send it to *A*. *A* decrypts all the received packets and then produce the partial result base on aggregation function.
3) The data transmission phase: each aggregator encrypts its calculated result and sends it to the upper level aggregator. The upper level aggregator decrypts all the received packets and aggregates them as a new aggregation result and then encrypts it again. Finally, the sink gets the aggregation result of the whole network.

As compared with first solution, HBH encryption is more efficient than previous approaches. It reduces the communication cost. However, HBH encryption has a serious flaw. It is vulnerable to attackers because their aggregated data is exposed in plaintext at the aggregator. An adversary can obtain some confidential information easily when an aggregator is compromised. Besides, another obvious drawback is that it requires three steps for aggregation, including decryption, aggregation, and re-encryption which instead increases the computation cost.

In order to achieve the efficient and secure data aggregation, we propose the end-to-end privacy preserving aggregation scheme, which hold both of the advantage of earlier two solutions. At the aggregator, we achieve to process encrypted data by using homomorphic encryption scheme and decrease communication cost.

# 3. Preliminaries

In this section we introduce some tools we used in our protocol. We first describe the concept of homomorphic encryption and then introduce the homomorphic encryption scheme we used for constructing our protocol.

## 3.1 Homomorphic Encryption Scheme

A homomorphic encryption scheme allows arithmetic operations to be performed on ciphertext. A homomorphic encryption scheme is useful when someone does not have decryption key but needs to fulfill some arithmetic operation on ciphertext. This method is consistent with our purpose. Let $E_k(.)$ be an encryption function and $D_k(.)$ be a decryption function. And then we define following operations on ciphertext.

- First operation $\oplus$:
  $E_k(m_1+m_2)=E_k(m_1)\oplus E_k(m_2)=c_1\oplus c_2$ where $c_1=E_k(m_1)$ and $c_2=E_k(m_2)$.
- Second operation $\otimes$:
  $E_k(rm)=E_k(m)\oplus E_k(m)\oplus\ldots\oplus E_k(m)=r\otimes E_k(m)$, where $r$ is a known constant.

In our protocol, we need additively homomorphic encryption schemes. There exists some encryption schemes with additive homomorphism. In this paper, we use *Privacy Homomorphisms* proposed by Domingo-Ferrer in [19] for our purpose. Although it is showed that this PH is insecure for some major parameter settings, it shows some reasons that such PH use for data aggregation scenarios in WSN is still reasonable secure and we can adopt the parameter setting discussed in [6].

## 3.2 Privacy Homomorphisms

In this section, we introduce a particular class of encryption transformations — PH, *Privacy Homomorphisms*. It can solve the problem of data aggregation without decrypting original messages in wireless sensor networks. First Privacy Homomorphism (PH) scheme is proposed by Rivest et al. In this paper, we use additive and multiplicative PH presented by Domingo-Ferrer [19]. The symmetric PH can be described as follows:

**Setup:**

*The public parameters*: positive integer $d$ > 2 and a large integer $w$. $w$ should have many small divisors and there should be many integers that can be inverted mod $w$.

*The secret parameters*: secret key $K = (x, q)$, $x \in Z_w$ such that $x^{-1}$ mod $w$ exists and a small divisor $q$ of $w$.

**Encryption:**

- Randomly split $a \in Z_q$ into secrets $a_1, a_2, .., a_d$ such that

  $a = a_1 + ... + a_d \bmod q = \sum_{j=1}^{d} a_j \bmod q$ and $a_j \in Z_w$ for all $j$.

- Compute $E_K(a) = (a_1 x \bmod w, a_2 x^2 \bmod w, ..., a_d x^d \bmod w)$

**Decryption:**

- Calculate the scalar product of the $j$-th coordinate by $x^{-j}$ mod $w$ to retrieve $a_j$ mod $w$ for all $j$.

- Compute $\sum_{j=1}^{d} a_j \bmod q$ to get the plaintext $a$.

In this case the set of plaintext is $T' = Z_q$. The set of ciphertext is $T = (Z_w)^d$. The set of plaintext operations is formed basically by addition, subtraction and multiplication in $T'$. The set of ciphertext operations is composed of addition, subtraction, defined as follows.

**Addition homomorphism:** They are done component-wise.

$$E_k(a) = \left(a_1 x \bmod w, a_2 x^2 \bmod w, \ldots, a_d x^d \bmod w\right)$$

$$E_k(b) = \left(b_1 x \bmod w, b_2 x^2 \bmod w, \ldots, b_d x^d \bmod w\right)$$

$$\begin{aligned}
&E_k(a+b) \\
&= \left((a_1 + b_1)x \bmod w, (a_2 + b_2)x^2 \bmod w, \ldots, (a_d + b_d)x^d \bmod w\right) \\
&= \left(a_1 x \bmod w, \ldots, a_d x^d \bmod w\right) + \left(b_1 x \bmod w, \ldots, b_d x^d \bmod w\right) \\
&= E_k(a) \oplus E_k(b)
\end{aligned}$$

# 4. Protocol Model and Background

Before describing our main protocol, we concentrate on interpreting a description of the protocol model and background.

## 4.1 Data Aggregation Model and Problem Definition

In a sensor network, the goal of aggregation is to compute those aggregation functions (like Sum, Maximum/Minimum, Average, Medium, Count) of the sensed readings on every sensor node. In this paper, the general sensor network is composed of many resource-limited sensor nodes and networks are illustrated in Figure 1. In our structure, there is a set $S = \{s_1, s_2, ..., s_n\}$ of *n* sensing nodes, each sensing node $s_i$ has sensed data value $v_i$. And there is a single base station *R* (or query server, sink node etc.), which is able to communicate with sensor nodes and has unlimited power and storage capability. Some of intermediate sensor nodes become aggregators (in this paper we also say cluster leader) $A = \{A_1, ..., A_k\}$. Due to large power consumption, when transmitting data packets, an aggregator is used to aggregate partial sensed data for reducing total communication cost and energy-saving.

The aggregation within sensor network is performed over an aggregation tree, which is the tree structure formed by the union of all the paths from the sensor nodes to the base station. The same as other data aggregation protocols, we assume the base station is the root of aggregation tree. There are multiple methods for constructing the aggregation tree, but we focus on providing security aspects of data aggregation.

A data aggregation function can represent $y = f(v_1, ..., v_n)$. In this paper, we focus on finding maximum/minimum in sensor network. Therefore,

$f(v_1, ..., v_n) = MAX\{v_1, ..., v_n\}$

## 4.2 Assumption

We assume that the base station *R* shares a unique key with every sensor node in the network for confidentiality communication. In addition, we assume *R* is unable to be compromised and it can authenticate its broadcast messages to all of sensor nods in aggregation tree [21].

For the data transmission, we also assume that there is a reliable transmission mechanism between nodes and it means that packets will not be loss when secure data aggregation procedure was afoot. For the key setup, we assume every sensing node has a common secret key $K_R = (x, q)$ shared with the base station and updates periodically. Additionally, there is a unique pairwise key shared between sensing nodes and the aggregator within a cluster, we can use proposed mechanism called random key distribution proposed in [15]. Besides, we assume the adversary has no knowledge about the WSNs and thus captures nodes randomly. *Finally, we assume hop-by-hop authentication between nodes.*

## 4.3 Key Setup for Encryption

In our protocol, we need two layer data encryption for secure computation in wireless sensor networks. The first layer is end-to-end encryption which wireless sensor networks to conceal the sensed data and aggregate data readings securely. Hence, the aggregators are unable to read the private data of sensing nodes. The second layer encryption is to establish a secure channel between the aggregator and sensor nodes. Because of the resource-constrained and storage-limited, we use an encryption transformation called *privacy homomorphisms* [19] to achieve our target for first layer encryption. We use another symmetric encryption scheme such as AES or RC5 for the second layer encryption. In the setup phase of our protocol, all of sensing nodes in the network need to agree on a secret key $K_R = (x, q)$ with base station privately. There contains several methods to achieve this job and we omit this issue in this paper.

## 4.4 Attacker Model

In this section, we describe the adversary's attempt. We first classify the adversary model.

**Semi-honest (Honest-but-Curious or Passive) Adversary:**

In this model, the attacker will conscientiously follow the prescribed protocol, but will try to learn or compute additional information during following the protocol. In other words, the target of an attacker is to compromise some nodes and read all messages in storage as well as eavesdropped in WSN.

**Malicious (Active) Adversary:**

In this model, the adversary deviates from the protocol in arbitrary ways. The purpose of this deviation can be several reasons, including learning more information from honest parties, modifying the result of the protocol, interfering the procedure of specified protocol.

**Collusive:**

We consider that any two participants (maybe two sensing nodes or one is sensing node and the other is aggregator) within cluster are collusive if they use their mutual secrets to derive the additional information.

**Non-collusive:**

On the other hand, we say two participants are non-collusive means that there are no two parties collude with others.

In this paper we do not consider the malicious adversary, we focus on defeating the attack from semi-honest and non-collusive adversary. In wireless sensor networks, the adversary can compromise a (small) $l \ (< n+k)$ fraction of sensors. We say that the adversary compromised a node means as long as it remains in control of sensor nodes, it can read all of contents and eavesdrops all incoming and outgoing messages. An adversary is interested in learning the private information of sensor nodes while remaining undetected. In addition, an adversary does not interfere with any communication over the network and modify sensed data on sensors it compromised. We assume adversary is unable to monitor and record all traffic and can only monitor incoming and outgoing communication of compromised nodes.

## 4.5 Requirement of Secure Data Aggregation

For secure data aggregation, our goal is to achieve end-to-end data privacy in a wireless sensor network. We must prevent a semi-honest adversary (eavesdropper) from obtaining any private information about sensor nodes. The following list the desired characteristics of a secure data aggregation.

**Correctness:** a correct aggregation of sensor data is desired. In this paper our purpose is to correctly obtain the maximum/minimum value of sensor networks with the constraint that no other sensors know the additional information of any individual sensor.

**Privacy:** for data confidentiality, there are two privacy goals. First, only the base station can learn about the final aggregation result. Second, each node only has knowledge about its private data after running data aggregation procedure. In another word, the normal neighbor nodes should not be able to know the private readings of other nodes and the secure data aggregation protocol should be able to defeat eavesdropper to reveal private data and ensure that the adversary can't deduce the plaintext. Besides, o

**Efficiency:** the purpose of data aggregation is to reduce communication overhead within whole network, thus reduce the power and resource consumption. Data aggregation can be achieved by in-network processing.

## 4.6 Notations

For clearly, we summarize the notation and symbols used in our protocol in Table 1.

Table 1: Notations

| Notations | Significance |
|---|---|
| $s$ | A sensor node (in here we refer to sensing nodes) |
| $A$ | An aggregator (or cluster leader) of a cluster. |
| $n$ | The total number of sensing nodes in the network. |
| $i$ | Sensor indices |
| $j$ | The indices of bit string |
| $l$ | The number of compromised nodes. |
| $s_i$ | A unique identifier of a sensor node. |
| $m$ | The bit length of sensing data. |
| $k$ | The total number of aggregator in the network. |
| $v_i$ | Sensor readings of node $s_i$. |
| $R$ | A base station of a wireless sensor network |
| $K_R$ | A common key shared between base station and all of sensing nodes |
| $K_{u,v}$ | A unique pair-wise key established between node $u$ and node $v$. |
| $r$, $r'$ | Random numbers |

| | |
|---|---|
| $E_{K_R}(.)$ | The encryption algorithm using shared secret key $K_R$ by Privacy Homomorhpim |
| $E'_{K_{u,v}}(.)$ | The symmetric encryption algorithm using secret pair-wise key $K_{u,v}$.(such as AES, RC5) |
| $\boldsymbol{c_i}$ | The ciphertext of $v_i$. |
| $\mathscr{M}$ | The message space of the encryption scheme |

# 5. The Main Protocol

In this section, we will introduce our main structure for finding maximum/minimum securely in wireless sensor networks. We first give an overview of the protocol and then present details of the protocol.

## 5.1 Protocol Overview

Our scheme is composed of three stages: the first stage is cluster-based aggregation tree formation, in this stage our goal is to construct cluster-based aggregation tree. One sensor node will be a cluster leader (or aggregator) which computes the maximum/minimum value over ciphertext within a cluster. The remaining members are sensing node which used to collect data for specific task and aggregation tree of cluster leaders is formed. The second stage is finding maximum/minimum within a cluster, in this phase we will calculate maximum/minimum of all the members within a cluster by a series of operations, then the result will pass to the upper layer cluster. The last stage is cluster data aggregate. All the aggregators become the members of the upper layer cluster in this phase. The following sections are details of three stages. For consistency, we use the aggregators to represent the cluster leaders in this paper.

## 5.2 Cluster-based Aggregation Tree Formation

The first step is to construct the aggregation tree for wireless sensor networks. The optimization of the aggregation tree structure is out of the scope of this paper. For concreteness, we describe the algorithm in [11]. In the sensor networks, the query server (or base station) broadcasts a query message “HELLO” which contains the tree construction information to other adjacent sensor nodes. Until each sensor node receives the query message, it elects itself as the cluster leader with some probability. If a node becomes a cluster leader, it will forward a query message to its neighbors as same as the query server. We say two nodes are neighbor means they can communicate with each other directly (one-hop); otherwise, it can decide to join one of the clusters in the sensor networks by responding a "JOIN" message which contains node's id to join. This procedure of cluster formation is illustrated in Figure 2.

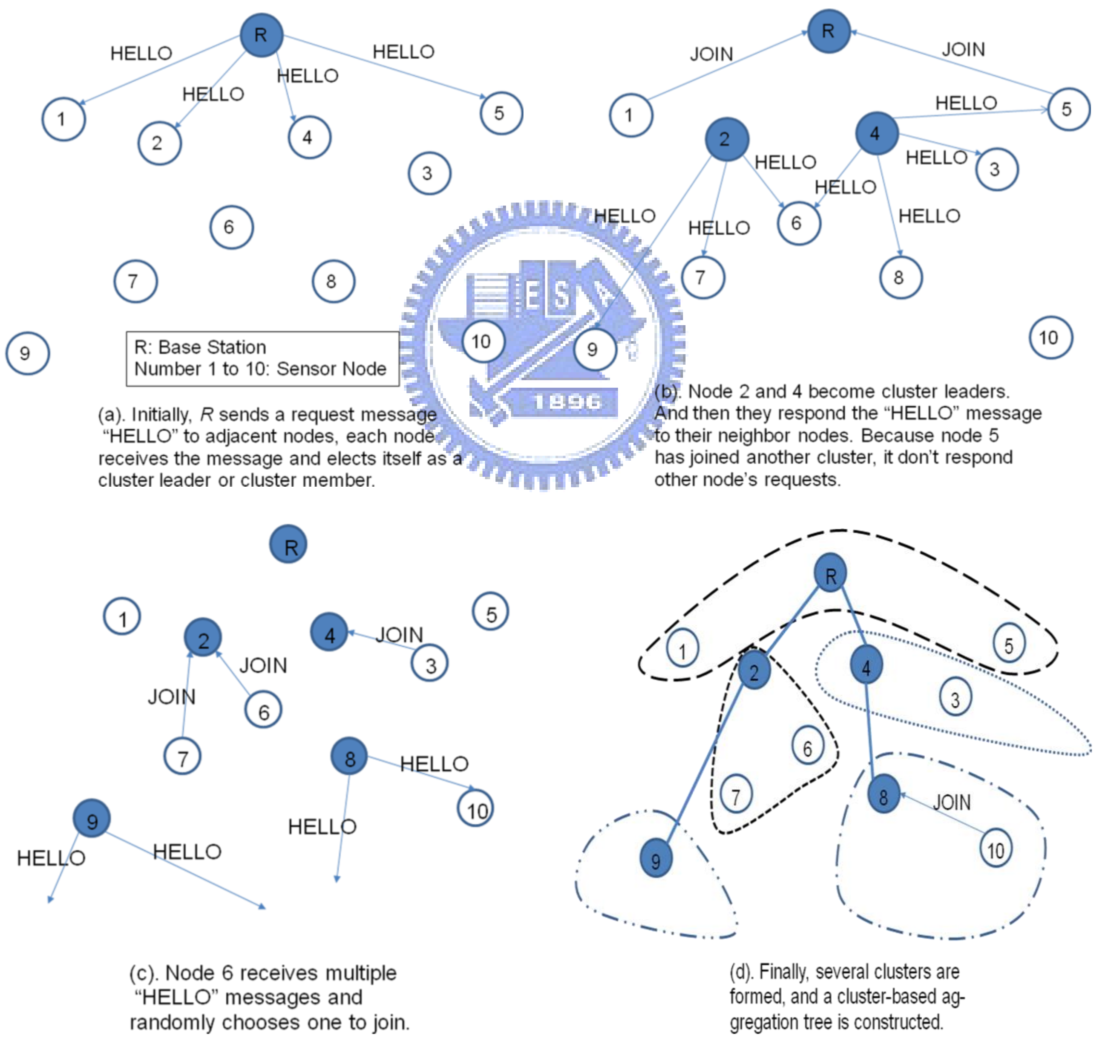

HELLO
R
HELLO
HELLO
HELLO
1
2
4
5
3
6
7
8
9
R: Base Station
Number 1 to 10: Sensor Node
10
9
(a). Initially, R sends a request message "HELLO" to adjacent nodes, each node receives the message and elects itself as a cluster leader or cluster member.
JOIN
R
JOIN
HELLO
1
2
4
5
HELLO
HELLO
3
HELLO
HELLO
HELLO
HELLO
6
7
8
10
(b). Node 2 and 4 become cluster leaders. And then they respond the "HELLO" message to their neighbor nodes. Because node 5 has joined another cluster, it don't respond other node's requests.
R
1
2
4
JOIN
5
3
JOIN
JOIN
6
7
8
HELLO
10
9
HELLO
HELLO
HELLO
(c). Node 6 receives multiple "HELLO" messages and randomly chooses one to join.
R
1
2
4
5
3
6
7
8
JOIN
10
9
(d). Finally, several clusters are formed, and a cluster-based aggregation tree is constructed.

**Figure 2. Cluster-based aggregation tree formation**

## 5.3 Finding Maximum/Minimum within Cluster

In this phase, our main idea is to provide secure data aggregation for finding maximum/minimum. We can divide our scheme into three steps. Three steps are *Encrypted Data Transmission, Decrypt then Aggregate, Verification.* The algorithm is illustrated in the following. $E_{K_R}$ and $D_{K_R}$ represent the encryption and the decryption algorithm where $K_R$ is the key. Let $|v|$ be the bits length of *v*. We use $v_{x,i}$ to represent the *i*-th bit of the $v_x$, where $v_x = v_{x,m} v_{x,m-1} \ldots v_{x,1}$. If we want to encrypt a value $a = \langle a_m, a_{m-1}, \ldots, a_1 \rangle$, we use $E(a)$ to denote $\langle E(a_m), E(a_{m-1}), \ldots, E(a_1) \rangle$.

- Aggregator *A* has a shared key $K_{A,i}$ with each sensor node $s_i$, respectively.
- Each sensor node has private sensing data, we denote $v_i$, where $m = |v_i|$
- Each sensor node has a common key $K_R = (x, q)$, which is also known to base station *R*, but the aggregator *A* does not know.
- Denote the number of sensing nodes in a cluster as *t*.

**Encrypted data transmission**

```
1    Encrypted data transmission
     1.1  Each sensor node $s_i$ encrypts its data reading $v_i$ by utilizing common
          key $K_R$, denote $E_{K_R}(v_i) = E_{K_R}(v_{i,m}), \ldots, E_{K_R}(v_{i,1})$
     1.2  Then each sensor node encrypts the message again by using shared
          secret key with A, denote $c_i = E'_{K_{A,i}}(E_{K_R}(v_{i,j}))$, where $1 \le i \le t$,
          $1 \le j \le m$.
     1.3  One of cluster member encrypts "-1" and then sends to the aggregator
          for using in next phase.
     1.4  Each sensing node sends its encrypted value to A respectively.
```

**Decrypt and Aggregate**

2 Decrypt and Aggregate

2.1 For each $j \in \{1,2,...,m\}$, *A* receives all the messages, decrypts the received messages from sensing nodes, and gets $E_{K_R}(v_{i,j})$ and then saves the value on its memory unit.

2.2 Choosing two parties $s_x$ and $s_y$ within a cluster. Then computing the following values via homomorphic encryption.

(a) $d_j = v_{x,j} - v_{y,j}$, $d_j' = v_{x,j} + v_{y,j} - 1$

(b) $e_j = r_j e_{j+1} + d_j$, $e_j' = r_j' d_j'$ where $e_{m+1} = 0$, $r_j, r_j' \in_R \mathcal{M}$.

(c) $f_j = e_j + e_j'$

**Verification**

3 Verification

3.1 *A* randomly sends $E_{K_R}(f)$ to one of sensing nodes $s_z$ in random order, where $f = \langle f_m, ..., f_1 \rangle$

3.2 $s_z$ decrypts the received message and computes the value *p*, then sends *p* to *A*.

3.3 *A* receives *p* and then knows which one is greater than another.

3.4 An aggregator *A* repeats (step 2.2 to step 3.3) until whole nodes are compared.

### 5.3.1 Details of Protocol

In this section we describe the details of our scheme. We use an example to illustrate. There are three parties within a cluster. One is aggregator, the remaining parties called node $s_x$ and node $s_y$ are sensing nodes which are used to collect sensitive data. Aggregator *A* compares the values $v_x, v_y$ which are collected by $s_x$ and $s_y$, respectively. We define the length of the sensed value is *m*, so if the sensed value which length is not sufficient *m*, the sensor node can prefix 0's to its value. We assume *A* has a shared key with every sensor node within the cluster. In here, we denote $K_{A,x}, K_{A,y}$. Moreover, we assume $K_R = (x, q)$ to be known to sensing nodes $s_x$ and $s_y$ and the base station.

In *Encrypted Data Transmission* phase, each sensor node ($s_x$ and $s_y$) bit-wisely encrypts its data reading $v_x, v_y$ by using common shared key $K_R$, denote $E_{K_R}(v_{x,m}),\ldots, E_{K_R}(v_{x,1}), E_{K_R}(v_{y,m}),\ldots, E_{K_R}(v_{y,1})$. Then these encrypted values will encrypt again by using shared key $K_{A,x}$ and $K_{A,y}$ with the aggregator. We denote $c_x = E'_{K_{A,x}}\left(E_{K_R}(v_{x,j})\right)$ and $c_y = E'_{K_{A,y}}\left(E_{K_R}(v_{y,j})\right)$ where $1 \le j \le m$. In addition, one of the sensing nodes encrypts the value "-1" by the *privacy homomorphisms* encryption scheme and sends to the aggregator. This encrypted value will be used in the following step, and we will explain why we need to do this later. Each sensor node sends the encrypted value to the aggregator.

In *Decrypt and Aggregate* phase, an aggregator receives messages from $s_x$ and $s_y$, and then decrypts messages by shared secret key $K_{A,x}, K_{A,y}$. An aggregator obtains $E_{K_R}(v_{x,m}),\ldots, E_{K_R}(v_{x,1})$, $E_{K_R}(v_{y,m}),\ldots, E_{K_R}(v_{y,1})$ and records it on its memory unit. Finally, the aggregator chooses two records to compute following values (for *i* = *m* to 1) via homomorphic encryption.

(a). $E_{K_R}(d_i) = E_{K_R}(v_{x,i} - v_{y,i}) = E_{K_R}(v_{x,i}) \oplus E_{K_R}(-v_{y,i})$

$E_{K_R}(d_i') = E_{K_R}(v_{x,i} + v_{y,i} - 1) = E_{K_R}(v_{x,i}) \oplus E_{K_R}(v_{y,i}) \oplus E_{K_R}(-1)$ where $E_{K_R}(-1)$ is given in *Encrypted Data Transmission* phase. Since the aggregator can't generate encryption of "-1" (otherwise the aggregator will know the secret parameter *x*), so we need to given the aggregator this encrypted value before computing this equation.

(b). $E_{K_R}(e_i)=E_{K_R}(r_i e_{i+1}+d_i)=r_i \otimes E_{K_R}(e_{i+1})\oplus E_{K_R}(d_i)$

$E_{K_R}(e_i')=E_{K_R}(r_i' d_i')=r_i' \otimes E_{K_R}(d_i')$ where $e_{m+1}=0$, and $r, r' \in_R \mathcal{M}$.

(c). $E_{K_R}(f_i)=E_{K_R}(e_i+e_i')=E_{K_R}(e_i)\oplus E_{K_R}(e_i')$

After an aggregator has completed the five equations, it will go to next phase.

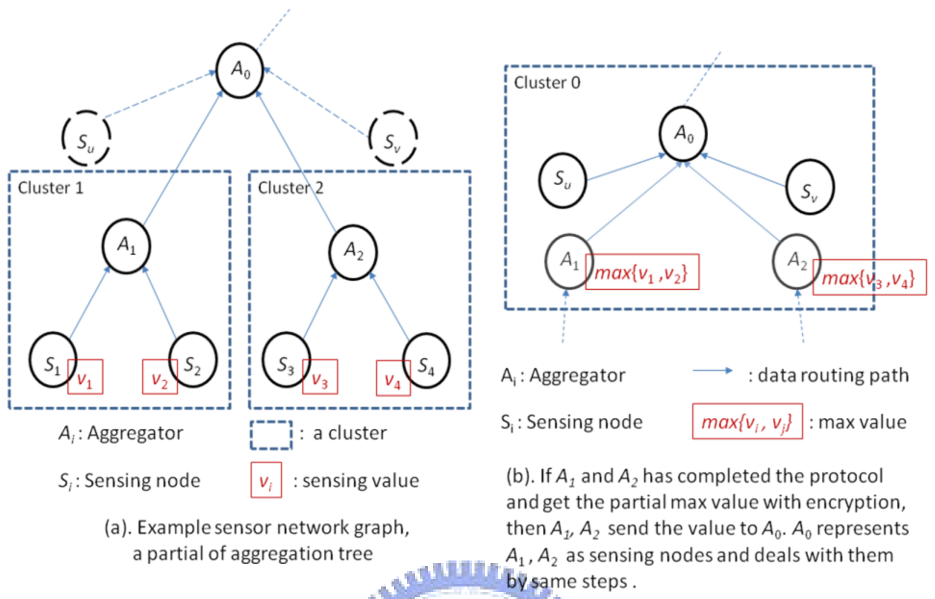
In *Verification* phase, the aggregator sends $E_{K_R}(f)$ to node $s_x$ or $s_y$ (assume $A$ sends to $s_x$) in random order, where $E_{K_R}(f)=\langle E_{K_R}(f_m),\ldots,E_{K_R}(f_1)\rangle$. Node $s_x$ decrypts the messages by common key $K_R$ and gets a vector $f'=\langle f'_m,\ldots,f'_1\rangle$ ($f'$ is random order of $f$), and then checks vector $f'$ then output $p$ by following statements.

$$p=\begin{cases} 1 & \text{, if there exists } f'_i=1 \text{ for } 1\le i\le m \\ 0 & \text{, if } f'_i=r'' \text{ for} \forall\, 1\le i\le m, r''\in_R M \\ -1 & \text{, if there exists } f'_i=-1 \text{ for } 1\le i\le m \end{cases}$$

Finally, $s_x$ sends $p$ to the aggregator to determinate which ciphertext is greater than another (if $p$ = 1, $v_x > v_y$; if $p$ = -1, $v_x < v_y$; if $p$ = 0, $v_x = v_y$). Then the aggregator keeps the greatest ciphertext in its memory unit and discards all remainder records. The comparison procedure is complete. If there exists another sensing node $s_z$ within cluster, the aggregator repeats the same procedure until all nodes has been compared. Numerical example is illustrated in Appendix A.

## 5.4 Cluster Data Aggregation

After finding maximum/minimum within a cluster, we need to aggregate data for all clusters. A common technique for data aggregation is to construct an aggregation tree. We implement our protocol on top of the cluster-based aggregation tree. Each aggregator forwards the derived maximum/minimum value within the cluster to the upper layer cluster and the upper layer aggregator represents the derived value received from the lower layer aggregator as another sensing value. We are illustrated in Figure 3 .



$A_0$
$S_u$
$S_v$
Cluster 1
Cluster 2
$A_1$
$A_2$
$S_1$
$S_2$
$S_3$
$S_4$
$v_1$
$v_2$
$v_3$
$v_4$
$A_i$: Aggregator
: a cluster
$S_i$: Sensing node
$v_i$ : sensing value
(a). Example sensor network graph, a partial of aggregation tree
Cluster 0
$max\{v_1, v_2\}$
$max\{v_3, v_4\}$
$A_i$: Aggregator
: data routing path
$S_i$: Sensing node
$max\{v_i, v_j\}$ : max value
(b). If $A_1$ and $A_2$ has completed the protocol and get the partial max value with encryption, then $A_1$, $A_2$ send the value to $A_0$. $A_0$ represents $A_1$, $A_2$ as sensing nodes and deals with them by same steps .

**Figure 3. Cluster data aggregation**

Figure 3(a) show the original structure, where $s_1$, $s_2$, $s_3$, $s_4$ are sensing nodes, $A_0$, $A_1$, $A_2$ are aggregators. Each sensing node $s_i$ has private data readings $v_i$. The cluster1 is formed with $A_1$, $s_1$, $s_2$ and cluster2 is formed with $A_2$, $s_3$, $s_4$. During the data aggregation, cluster1 will perform *finding max/min* protocol and obtain max$\{v_1, v_2\}$. Cluster2 will get max$\{v_3, v_4\}$ by the same token. Next, we fulfill cluster data aggregation and illustrate it in Figure 3(b).

As we said earlier, the aggregators $A_1$, $A_2$ get the partial maximum value within a cluster, and then they send partial maximum/minimum value to the aggregator $A_0$. Further, $A_0$ is the aggregator and $A_1$, $A_2$ become cluster members in Cluster0. $A_0$ completes data comparison and obtains partial maximum/minimum by the same steps. Because of the cluster-based tree aggregation architecture, there must exists sensing nodes within a cluster. Therefore, our protocol works correctly.

# 6. Security Analysis

In this section, we briefly discuss correctness and privacy of our protocol.

## 6.1 Correctness

In terms of correctness of our protocol, our purpose is to let the base station can acquire the correct maximum/minimum value of the sensor networks. In order to achieve the goal, we concentrate on the *Decrypt and Aggregate* phase. For simplification, we assume there are two sensing nodes within a cluster which identifier are $s_x$, $s_y$ with private readings $v_x$, $v_y$, respectively. Considering the following three cases:

Case 1: if $v_x > v_y$, we let $l$ be the index of the first different bit of $v_x$ and $v_y$ from the most significant bit. We obviously discover that the value of bit index $j$ of $v_x$ and $v_y$ where

- $l+1 \le j \le m$: $e_j = d_j = 0$ and $d_j' = 1$ or $-1$ such that $e_j'$ is always a random number. Therefore, the derived value $f_j$ is a random number too.
- $j = l$: $e_j = d_j = 1$ and $e_j' = d_j' = 0$. Therefore, $f_j = e_j + e_j' = 1$.
- $1 \le j \le l-1$: $e_j$ is always a random number such that the derived value $f_j$ is a random number too.

Therefore, we deduce the identifier value $p = 1$.

Case 2: if $v_x < v_y$, as same as case 1, we can list the situations of bit index $j$ as follows:

- $l+1 \le j \le m$: The derived value $f_j$ is a random number.
- $j = l$: $e_l = d_l = -1$ and $e_l' = d_l' = 0$. Therefore, $f_l = e_l + e_l' = -1$.
- $1 \le j \le l-1$: The derived value $f_j$ is a random number too.

Therefore, we deduce the identifier value $p$ = -1.

Case 3: if $v_x = v_y$, it means that all of bits of $v_x$ and $v_y$ are same, such that $e_j = d_j = 0$, $d_j' = 1 \text{ or } -1$ and $e_j'$ is always a random number, and thus $f_j = e_j + e_j'$ is also a random number too, where $1 \le j \le m$. Therefore, we deduce the identifier value $p$ = 0.

After the protocol, we can determine which one is greater than other accurately. In ideal situations when there is no data loss in the sensor network, base station should get accurate aggregation results.

## 6.2 Privacy

Assume that the involved parties are *semi-honest* and *non-collusive*. We discuss the data confidentiality of our scheme into two conditions. For simplification, we consider there are two sensing nodes $s_x$ and $s_y$ with private data $v_x$ and $v_y$ and an aggregator $A$ within the same cluster. First, the compromised node is one of sensing nodes, we assume the compromised node is $s_y$ ($s_x$ and $s_y$ are symmetric). The attacker can obtain all incoming and outgoing messages of $s_y$. Additionally, key information and private data of node $s_y$ are revealed. Through the protocol, the messages of $s_y$ can obtain from the aggregator is $E_{K_R}(f')$.We briefly discuss why the adversary can't acquire extra information. Let $l$ be index of the first different bit of $v_x$ and $v_y$. We see that if $v_x > v_y$ or $v_x < v_y$ , then $f'_j$ is uniformly distributed in $\mathcal{M}$, for all index $j \neq l$ and $f'_l$ = 1 ($v_x > v_y$), $f'_l$ = -1 ($v_x < v_y$) for index $j = l$. By $s_y$'s view, it can't get any bit information of $s_x$ because the $f'$ is in random order. Besides, it is not able to know which one is bigger than other since it don't know $A$ how to compare two values. Therefore, no other information leak to the compromised sensing node.

Second, the compromised node is the aggregator $A$. According to the procedure of our protocol, the adversary can receive the messages transmitted from sensing nodes. All the messages are encrypted by *privacy homomorphism*. Since our protocol is secure base on security of *privacy homomorphism*, we can construct another ADV' to break the *Privacy Homomorphism* scheme if there exist an attacker ADV break our protocol.

We brief discuss the security of *Privacy Homomorphism*. For a fixed number$\alpha$ of known cleartext-ciphertext pairs, the probability of randomly guessing the right key can be made small. In addition, there is only a small probability that a ciphertext decrypts to the same cleartext using two different keys. Combining both results, we consider that if the aggregator is compromised by an attacker, the attacker is hard to derive the secret value with received ciphertext from $s_x$ and $s_y$ since the attacker randomly guessing the right key. We show that $v_x$ is kept secret from $A$ and $s_y$, and $v_y$ is kept secret form $A$ and $s_x$ after the protocol.

However, above two cases are unrealistic situations in wireless sensor networks, and we show that even if the adversary compromised a small portion of nodes including the aggregator and sensing nodes, it only effect in *localizing* the possible damage. We see that if the adversary captures an aggregator $A$ and another sensing node $s_z$ ($s_z$ and $A$ are not in the same cluster), the private information of the cluster including the aggregator $A$ are revealed. But the nodes out of the range of this cluster only the private data of compromised nodes are exposed. For a realistic situation in wireless sensor networks, an adversary is unable to monitor all communication of whole networks. It only monitors incoming and outgoing traffic and private data of compromised nodes. Therefore, we restrict the security impact of nodes compromise.

# 7. Overhead

In this section, we evaluate the computation and communication cost for each node. We use bit size of the messages required for evaluating communication cost and number of operations for computation cost. We first consider the case of sensing node. However, we ignore some light-weight operations which include pseudo-random function. For convenience, the following notations are used for analysis.

- $C$ : the ciphertext space of privacy homomomrphism.
- $d$ : the number of division of PH. (in [6], they suggest d = 2~4)
- $t$ : the average number of cluster member.
- $C_{sym}$: the computation cost of symmetric encryption. (such as AES)
- $C_{mul}$: the computation cost of multiplication. (modulus operation)
- $C_{add}$: the computation cost of addition. (modulus operation)

First, we evaluate the messages overhead of ciphertext. We consider the ciphertext of PH. Because each sensitive data are encrypted bitwisely, and each encryption is split into $d$ pieces. The overhead of encrypted messages is ($m \cdot d \cdot \log C$). Through the protocol, each sensing node sends one ciphertext in *Encrypted Data Transmission* phase and then responds to the aggregator the value $p$ in *Verification* phase. Therefore, the communication cost of sensing node is ($md \log C$)+ $|p|$. Next, we evaluate the computation cost in a sensing node. For Domingo-Ferrer Privacy Homomorphism scheme, the encryption needs 2$d$-1 multiplication (modular operation) for transforming $d$ partitions to ciphertext. Therefore, the total cost of PH encryption is $(2d\text{-}1)C_{mul}$ . For the decryption of PH, it needs 2$d$-1 multiplication (modulus operation) for retrieving the original partitions and ($d$-1) addition (modulus operation) for retrieving the original messages. The total cost of decryption is $(2d\text{-}1)C_{mul}$ + $(d\text{-}1)C_{add}$. Through our protocol, a sensing node must encrypt data in *Encrypted data*

*Transmission*. In addition, the bit length of sensing data is $m$ and we assume the symmetric encryption used for hop-by-hop encryption requires $C_{sym}$. Therefore, the encryption cost $m(2d\text{-}1)C_{mul} + C_{sym}$ . Besides, it must decrypt one ciphertext received from the aggregator in average case, the cost is $m[(2d\text{-}1)C_{mul} + (d\text{-}1)C_{add}]$. Therefore, the computation cost of one sensing node is $m[(4d\text{-}2)C_{mul} + (d\text{-}1)C_{add}] + C_{sym}$.

In the following, we consider the aggregator. We assume there are $t$ members within a cluster in average case. An aggregator must decrypt the ciphertext $C_{sym}$ and compute ($t$-1) times for calculating the partial result and send $t$-1 encrypted messages to members in *Decrypt and Aggregate* phase. After finding the maximum/minimum value within a cluster, the aggregator re-encrypts the result by symmetric encryption and then sends the calculated partial result to upper layer cluster for next aggregation in the end. Therefore, total communication of the aggregator is $t\,(md\log C)$. For an aggregator, it costs five addition (modulus operation) and two multiplication (modulus operation) for comparing two encrypted data. Additionally, we need $t$-1 times comparisons for finding maximum/minimum. Therefore, total computation cost of aggregators is $(t\text{-}1)(5C_{add} + 2C_{mul}) + (t\text{+}1)C_{sym}$. The summarization of computation cost and communication cost is list in Table 2.

Table 2. The communication and computation cost of each node

| | **Communication Cost** | **Computation Cost** |
|---|---|---|
| **Sensing node** | $(md\log C) + \lvert p\rvert$ | $m[(4d\text{-}2)C_{mul} + (d\text{-}1)C_{add}] + C_{sym}$ |
| **Aggregator** | $t\,(md\log C)$ | $(t\text{-}1)(5C_{add}+2C_{mul})+(t\text{+}1)C_{sym}$ |

## 7.1 Comparison

In this section, we compare our scheme to another secure comparison scheme proposed by Acharya et al. [8]. In [8], they use an order preserving encryption scheme (OPES). This scheme allows any comparison operation to be applied on encrypted data directly. Therefore, the sensing nodes only encrypt sensitive data and send to the aggregator directly, and the aggregator only need to compare the ciphertext received from sensing nodes.

We briefly compare two schemes by concrete measurements from [8] and [22]. The Crossbow's MICA2 Motes are one candidate for a platform. For the encryption transformations of OPES and $PH_{d=2}$ (privacy homomorphism with parameter $d$=2), they measured execution times in terms of clock cycles for encryption and decryption. We find the total cost of OPES encryption is about 1800 clock cycles for 4 bytes operands by some measurements presented in [8], and the total cost of $PH_{d=2}$ encryption is about 5700 clock cycles by some measurements presented in [22]. In addition, our scheme needs extra energy consumption for symmetric encryption and the decryption at a sensing node. Therefore, the energy consumption of our scheme is more than OPES scheme. Take security issue into account, we focus on two kinds of attacks, one is *node compromise attack*, another is *eavesdropping*. Our scheme can defeat against the node compromise attack in some situations, but the OPES scheme is unable to resist such attack. In the following, we summarize the comparison of two schemes in Table 3.

Table 3. The comparison of OPES scheme and our scheme

| | | OPES | Our scheme |
|---|---|---|---|
| **Energy Consumption** | **Aggregator** | **Efficient** | **Applicable** |
| | **Sensing node** | **Efficient** | **Applicable** |
| ***Node Compromise Attack** | **Aggregator** | **No** | **Yes** |
| | **Sensing node** | **No** | **Yes** |
| | **Both** | **No** | **No** |
| **Eavesdropping Attack** | | **Yes** | **Yes** |

***Node compromise attack: row "Aggregator" means the compromised nodes are aggregators; row "Sensing node" means the compromised nodes are sensing nodes; row "Both" means the compromised nodes include aggregators and sensing nodes.**

**Notation: Yes: satisfied No: not satisfied**

# 8. Conclusions and Future Work

In this paper, we first employ the cryptographic technique which used to solve the problem of secure comparison for finding maximum/minimum in wireless sensor networks. In our scheme, we use the symmetric privacy homomorphism purposed by Domingo-Ferrer. It is feasible to process operations over the ciphertext in a wireless sensor network. Our protocol has some properties as follows:

- Providing end-to-end privacy: if the adversaries are semi-honest and only compromise the same kind of nodes (aggregator or sensing node), no private information of uncompromised nodes is disclosed. If the adversary compromise aggregators and sensing nodes, it only effect in *localizing* the possible damage. In terms of security, the result of our scheme is better than the preceding scheme in [8].
- Efficiency: our scheme can work with data aggregation and reduce energy consumption.

We have proved that our scheme is correct and provide data privacy. For the future work, there are two purposes we desire to achieve. First, we want to expand our scheme to defeat against the malicious adversary. Second, we expect to minimize the damage of node compromised attack.

# Reference

[1]. D. Liu, P. Ning. Security for wireless sensor networks: introduction. Advances in Information Security, Volume28, pp. 1-7, Springer, 2007.

[2]. D. Culler, D. Estrin, and M. Srivastava, Overview of Sensor Networks, *IEEE Computer*, 37(8), pp. 41-46, August 2004.

[3]. S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: a tiny aggragation service for ad-hoc sensor networks. In *Proceedings of the fifth Annual Symposium on Operation Systems Design and Implementation* (OSDI’02), 2002.

[4]. B. Przydatek, D. Song, and A. Perrig. SIA: Secure information aggregation in sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems* (SenSys’03), pp. 255-265, ACM Press, 2003.

[5]. Y. Yang, X. Wang, S. Zhu, and G. Cao. SDAP: A secure hop-by-hop data aggregation protocol for sensor networks. In *Proceedings of the 7thACM International Symposium on Mobile Ad Hoc Networking and Computing* (MobiHoc’06), pp. 356-367, ACM Press, 2006.

[6]. J. Girao, M. Schneider, and D. Westhoff. CDA: Concealed data aggregation in wireless sensor networks. In *Proceedings of the ACM Workshop on Wireless Security* (Wise’04), 2004.

[7]. C. Castelluccia, E. Mykletun, and G.. Tsudik. Efficient aggregation of encrypted data in wireless sensor networks. In *Proceedings of the Second Annual International Conference on Mobile and Ubiquitous Systems*, pp. 109-117, IEEE Press, 2005.

[8] M. Acharya, J Girao, and D. Westhoff. Secure comparison of encrypted data in wireless sensor networks. In *Proceeding of the Third International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks* (WiOpt’05), pp. 47-53, IEEE Press, 2005.

[9]. P. Jadia and A. Mathuria. Efficient secure aggregation in sensor networks. In *Proceddings of the 11th International Conference on High Performance Computing* (HiPC'04), Lecture Notes in Computer Science 3296, pp. 40-49, Springer, 2004.

[10]. H. Chan, A. Perrig, and D. X. Song. Secure hierarchical in-network aggregation in sensor networks. In *Proceedings of the 13th ACM Conference on Computer and Communications Security* (CCS'06), pp. 278-287, ACM Press, 2006.

[11]. W. He, X. Liu, H. Nguyen, K. Nahrstedt, and T. F. Abdelzaher. PDA: Privacy-preserving data aggregation in wireless sensor networks. In *Proceedings of the 26th IEEE International Conference on Computer Communication* (INFOCOM'07), pp. 2045-2053, IEEE Press, 2007.

[12]. D. Liu and P. Ning, Establishing pairwise keys in distributed sensor networks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security* (CCS'03), pp. 52-61, ACM Press, 2003.

[13]. C. Bekara, M. Laurent-Maknavicius, and K. Bekara. SAPC: A secure aggregation protocol for cluster-based wireless sensor networks. In *Proceedings of the Third International Conference* on *Mobile Ad-Hoc and Sensor Networks* (MSN'07), Lecture Notes in Computer Science 4864, pp. 784-798, Springer, 2007.

[14]. K. Kifayat, M. Merabti, Q. Shi, and D. Llewellyn-Jones. Applying secure data aggregation techniques for a structure and density independent group based key management protocol. In *Proceedings of the Third International Symposium on Information and Assurance and Security* (IAS'07), pp. 44-49, IEEE Press, 2007.

[15] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security* (CCS'02), pp. 41-47, ACM Press, 2002.

[16] C. K. Chu, and W. G.. Tzeng. Conditional Oblivious Cast Schemes. In *Proceedings of the 9th International Conference on Theory and Practice of Public Key*

*Cryptography* (PKC'06), Lecture Notes in Computer Science 3958, pp. 443-457, Springer, 2006.

[17] C. Intanagonwiwant, D. Estrin, R. Govindan, and J. Heidemann. Impact of network density on data aggregation in wireless sensor networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems* (ICDCS'02), pp. 457-458, IEEE Press, 2002.

[18] B. Krishnamachari, D. Estrin, and S. Wicker. The impact of data aggregation in wireless sensor networks. In *Proceedings of the 22nd International Conference on Distributed Event-Based Systems, Workshops* (ICDCSW'02), pp. 575-578, IEEE Press, 2002.

[19] J. Domingo-Ferrer. A provably secure additive and multiplicative privacy homomorphism. In *Proceedings of the 5th International Conference on Information Security* (ISC'02), Lecture Notes in Computer Science 2433, pp. 471-483, Springer, 2002.

[20] K. Yuen, B. Li, B. Liang. Distributed minimum energy data gathering and aggregation in sensor networks. In *Proceedings of 2006 IEEE International Conference on Communications* (ICC'06), pp. 3536-3541, IEEE Press, 2006.

[21] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar. SPINS: Security protocols for sensor networks. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking* (MOBICOM'01), pp. 189-199, ACM Press, 2001.

[22] D. Westhoff, J. Girao, and M. Acharya. Concealed data aggregation for reverse multicast traffic in sensor networks: encryption, key distribution, and routing adaptation. In *Proceedings of 2005 IEEE Transaction on Mobile Computing* (ICC'05), pp. 1417-1431, IEEE Press, 2006

# Appendix A

In the following, we give an example for explanation.

## Numerical Example for Secure Comparison

For simplification, this example focuses on aggregating the encrypted data. Assume there are two participants $s_x$ and $s_y$ with private readings $v_x$ and $v_y$, respectively. We will compare two data in encrypted form with following parameters:

For illustration, we choose unrealistic small values:
*Public parameters:* $d$ = 2, a modulus $g$ = 28
*Secret parameters*: $K = (x, q)$ , $x$ = 3 and $q$ = 7
The public aggregation function MAX is $f(v_x, v_y) = max\{v_x, v_y\}$

**Case 1**: $v_x > v_y$ , $v_x = 5 = 101_2 = X_2X_1X_0$, $v_y = 3 = 011_2 = Y_2Y_1Y_0$

$s_x$ and $s_y$ compute the ciphertext bit-wisely $v'_x = X'_2\, X'_1\, X'_0\, , v'_y = Y'_2\, Y'_1\, Y'_0$

$$\begin{aligned}
X'_2 &= E_{(3,7)}(1) = E_{(3,7)}(4,4) = (12,8) \quad & Y'_2 &= E_{(3,7)}(0) = E_{(3,7)}(1,6) = (3,26) \\
X'_1 &= E_{(3,7)}(0) = E_{(3,7)}(3,4) = (9,8) \quad & Y'_1 &= E_{(3,7)}(1) = E_{(3,7)}(3,12) = (9,24) \\
X'_0 &= E_{(3,7)}(1) = E_{(3,7)}(6,2) = (18,18) \quad & Y'_0 &= E_{(3,7)}(1) = E_{(3,7)}(5,3) = (15,27)
\end{aligned}$$

and sends to the aggregator $AG$ with additional information (the ciphertext of “1” (24,7) ).

*AG* computes

$$d_2 = X'_2 - Y'_2 = (12,8) - (3,26) = (9,10)$$

$$d'_2 = X'_2 + Y'_2 - 1 = (12,8) + (3,26) - (24,7) = (19,27)$$

$AG$ chooses random numbers $r_2 = 3$, $r'_2 = 2$ and $e_3 = (0,0)$

$$e_2 = r_2 e_3 + d_2 = 3 \cdot (0,0) + (9,10) = (9,10)$$

$$e'_2 = r'_2\, d'_2 = 2 \cdot (19,27) = (10,26)$$

$$f'_2 = e_2 + e'_2 = (9,10) + (10,26) = (19,8)$$

$d_1 = X'_1 - Y'_1 = (9,8) - (9,24) = (0,12)$
$d'_1 = X'_1 + Y'_1 - 1 = (9,8) + (9,24) - (24,7) = (22,25)$
*AG* chooses random numbers $r_1 = 5, r'_1 = 3$ and $e_2 = (9,10)$
$e_1 = r_1 e_2 + d_1 = 5 \cdot (9,10) + (0,12) = (17,6)$
$e'_1 = r'_1\, d'_1 = 3 \cdot (22,25) = (10,19)$
$f'_1 = e_1 + e'_1 = (17,6) + (10,19) = (27,25)$

$d_0 = X'_0 - Y'_0 = (18,18) - (15,27) = (3,19)$
$d'_0 = X'_0 + Y'_0 - 1 = (18,18) + (15,27) - (24,7) = (9,10)$
*AG* chooses random numbers $r_0 = 4$, $r'_0 = 3$ and $e_1 = (17,6)$
$e_0 = r_0 e_1 + d_0 = 4 \cdot (17,6) + (3,19) = (15,15)$
$e'_0 = r'_0\, d'_0 = 3 \cdot (9,10) = (27,2)$
$f'_0 = e_0 + e'_0 = (15,15) + (27,2) = (14,17)$

Then *AG* transmits the final $f' = f'_2 f'_1 f'_0$ to $s_x$ or $s_y$.

The sensing node decrypts $f'$ by computing

$f_2 = D_{(3,7)}(f'_2) = D_{(3,7)}(19,8) = (19 * 19 \bmod 28, 8 * 19 * 19 \bmod 28) = (25,4)$
$= (25 + 4) \bmod 7 = 1$
$f_1 = D_{(3,7)}(f'_1) = D_{(3,7)}(27,25) = (27 * 19 \bmod 28, 25 * 19 * 19 \bmod 28) = (9,9)$
$= (9 + 9) \bmod 7 = 4$
$f_0 = D_{(3,7)}(f'_0) = D_{(3,7)}(14,17) = (14 * 19 \bmod 28, 17 * 19 * 19 \bmod 28) = (14,5)$
$= (14 + 5) \bmod 7 = 5$

The sensing node finds the value 1 in $f_2$. Therefore, it sets $p = 1$ and sends $p$ to *AG*. Finally, *AG* receives the $p$ and discards the ciphertext of $v_y$. In the same way, if $v_x < v_y$, the sensing node sets $p = -1$ and then *AG* discards the ciphertext of $v_x$.

**Case 2**: $v_x = v_y$ , $v_x = 3 = 011_2 = X_2X_1X_0$, $v_y = 3 = 011_2 = Y_2Y_1Y_0$

$s_x$ and $s_y$ compute the ciphertext bit-wisely $v'_x = X'_2\, X'_1\, X'_0$, $v'_y = Y'_2\, Y'_1\, Y'_0$

$X'_2 = E_{(3,7)}(0) = E_{(3,7)}(3,4) = (9,8)$ $Y'_2 = E_{(3,7)}(0) = E_{(3,7)}(1,6) = (3,26)$
$X'_1 = E_{(3,7)}(1) = E_{(3,7)}(4,4) = (12,8)$ $Y'_1 = E_{(3,7)}(1) = E_{(3,7)}(3,12) = (9,24)$
$X'_0 = E_{(3,7)}(1) = E_{(3,7)}(6,2) = (18,18)$ $Y'_0 = E_{(3,7)}(1) = E_{(3,7)}(5,3) = (15,27)$

and sends to *AG* with additional information (the ciphertext of “1” (24,7) ).

*AG* computes

$$d_2 = X'_2 - Y'_2 = (9,8) - (3,26) = (6,10)$$
$$d'_2 = X'_2 + Y'_2 - 1 = (9,8) + (3,26) - (24,7) = (16,27)$$
*AG* chooses random numbers $r_2 = 3$, $r'_2 = 2$ and $e_3 = (0,0)$
$$e_2 = r_2 e_3 + d_2 = 3 \cdot (0,0) + (6,10) = (6,10)$$
$$e'_2 = r'_2\, d'_2 = 2 \cdot (16,27) = (4,26)$$
$$f'_2 = e_2 + e'_2 = (6,10) + (4,26) = (10,8)$$

$$d_1 = X'_1 - Y'_1 = (12,8) - (9,24) = (3,12)$$
$$d'_1 = X'_1 + Y'_1 - 1 = (12,8) + (9,24) - (24,7) = (25,25)$$
*AG* chooses random numbers $r_1 = 5$, $r'_1 = 3$ and $e_2 = (6,10)$
$$e_1 = r_1 e_2 + d_1 = 5 \cdot (6,10) + (3,12) = (5,6)$$
$$e'_1 = r'_1\, d'_1 = 3 \cdot (25,25) = (19,19)$$
$$f'_1 = e_1 + e'_1 = (5,6) + (19,19) = (24,25)$$

$$d_0 = X'_0 - Y'_0 = (18,18) - (15,27) = (3,19)$$
$$d'_0 = X'_0 + Y'_0 - 1 = (18,18) + (15,27) - (24,7) = (9,10)$$
*AG* chooses random numbers $r_0 = 4$, $r'_0 = 3$ and $e_1 = (5,6)$
$$e_0 = r_0 e_1 + d_0 = 4 \cdot (5,6) + (3,19) = (23,15)$$
$$e'_0 = r'_0\, d'_0 = 3 \cdot (9,10) = (27,2)$$
$$f'_0 = e_0 + e'_0 = (23,15) + (27,2) = (22,17)$$

Then *AG* transmits the final $f' = f'_2 f'_1 f'_0$ to $s_x$ or $s_y$.

The sensing node decrypts $f'$ by computing

$$f_2 = D_{(3,7)}(f'_2) = D_{(3,7)}(10,8) = (10 * 19 \bmod 28, 8 * 19 * 19 \bmod 28) = (22,4)$$
$$= (22 + 4) \bmod 7 = 5$$
$$f_1 = D_{(3,7)}(f'_1) = D_{(3,7)}(24,25) = (24 * 19 \bmod 28, 25 * 19 * 19 \bmod 28) = (8,9)$$
$$= (8 + 9) \bmod 7 = 3$$
$$f_0 = D_{(3,7)}(f'_0) = D_{(3,7)}(22,17) = (22 * 19 \bmod 28, 17 * 19 * 19 \bmod 28) = (26,5)$$
$$= (26 + 5) \bmod 7 = 3$$

The sensing node finds that there are no 1 or -1 in $f$ ($f = f_2 f_1 f_0$). Therefore, it sets $p = 0$ and sends $p$ to *AG*. Finally, *AG* receives the $p$ and discards one of ciphertext arbitrary.