An Attack on Privacy Preserving Data Aggregation Protocol for Wireless Sensor Networks

Jaydip Sen[1] and Subhamoy Maitra[2]

[1] Innovation Labs, Tata Consultancy Services Ltd.,
Bengal Intelligent Park, Salt Lake Electronic Complex, Kolkata 700 091, India
jaydip.sen@acm.org

[2] Applied Statistics Unit, Indian Statistical Institute,
203 B T Road, Kolkata 700 108, India
subho@isical.ac.in

Abstract. In-network data aggregation in Wireless Sensor Networks (WSNs) provides efficient bandwidth utilization and energy-efficient computing. Supporting efficient in-network data aggregation while preserving the privacy of the data of individual sensor nodes has emerged as an important requirement in numerous WSN applications. For privacy-preserving data aggregation in WSNs, He et al. (INFOCOM 2007) have proposed a Cluster-based Private Data Aggregation (CPDA) that uses a clustering protocol and a well-known key distribution scheme for computing an additive aggregation function in a privacy-preserving manner. In spite of the wide popularity of CPDA, it has been observed that the protocol is not secure and it is also possible to enhance its efficiency. In this paper, we first identify a security vulnerability in the existing CPDA scheme, wherein we show how a malicious participant node can launch an attack on the privacy protocol so as to get access to the private data of its neighboring sensor nodes. Next it is shown how the existing CPDA scheme can be made more efficient by suitable modification of the protocol. Further, suitable modifications in the existing protocol have been proposed so as to plug the vulnerability of the protocol.

Keywords: Wireless sensor network, privacy, data aggregation, cluster-based private data aggregation (CPDA), key distribution, colluding attack, malicious node.

1 Introduction

In recent years, wireless sensor networks (WSNs) have drawn considerable attention from the research community on issues ranging from theoretical research to practical applications. Special characteristics of WSNs, such as resource constraints on energy and computational power and security have been well-defined and widely studied [2][12]. What has received less attention, however, is the critical privacy concern on information being collected, transmitted, and analyzed in a WSN. Such private and sensitive information may include payload data collected by sensors and transmitted through the network to a centralized data processing server. For example, a patient's blood pressure, sugar level and other vital signs are usually of critical privacy concern when monitored by a medical WSN which transmits the data to a remote hospital or doctor's office. Privacy concerns may also arise beyond data content and may focus on context information such as the location of a sensor initiating data communication. Effective countermeasure against the disclosure of both data and context-oriented private information is an indispensable prerequisite for deployment of WSNs in real-world applications.

Privacy protection has been extensively studied in various fields related to WSNs such as wired and wireless networking, databases and data mining. Nonetheless, the following inherent features of WSNs introduce unique challenges for privacy preservation in WSNs, and prevent the existing techniques from being directly transplanted: (i) *Uncontrollable environment*: Sensors may have to be deployed to an environment uncontrollable by the defender, such as a battlefield, enabling an adversary to launch physical attacks to capture sensor nodes or deploy counterfeit ones. As a result, an adversary may retrieve private keys used for secure communication and decrypt any communication eavesdropped by the adversary. (ii) *Sensor-node resource constraints*: battery-powered sensor nodes generally have severe constraints on their ability to store, process, and transmit the sensed data. As a result, the computational complexity and resource consumption of public-key ciphers is usually considered unsuitable for WSNs. (iii) *Topological constraints*: the limited communication range of sensor nodes in a WSN requires multiple hops in order to transmit data from the source to the base station. Such a multi-hop scheme demands different nodes to take diverse traffic loads. In particular, a node closer to the base station (i.e., data collecting and processing server) has to relay data from nodes further away from base station in addition to transmitting its own generated data, leading to higher transmission rate. Such an unbalanced network traffic pattern brings significant challenges to the protection of context-oriented privacy information. Particularly, if an adversary holds the ability of global traffic analysis, observing the traffic patterns of different nodes over the whole network, it can easily identify the sink and compromise context privacy, or even manipulate the sink node to impede the proper functioning of the WSN.

The unique challenges for privacy preservation in WSNs call for the development of effective privacy-preserving techniques. Supporting efficient in-network data aggregation while preserving data privacy has emerged as an important requirement in numerous wireless sensor network applications [1][4][8][9][13]. As a key approach to fulfilling this requirement of private data aggregation, *concealed data aggregation* (CDA) schemes have been proposed in which multiple source nodes send encrypted data to a sink along a *converge-cast tree* with aggregation of cipher-text being performed over the route [1][3][4][8][11][13].

He et al. have proposed a *cluster-based private data aggregation* (CPDA) scheme in which the sensor nodes are randomly distributed into clusters [9]. The cluster leaders carry out aggregation of data received from the cluster member nodes. The data communication is secured by using a shared key between each pair of communicating nodes for the purpose of encryption. The aggregate function leverages algebraic properties of the polynomials to compute the desired aggregate value in a cluster. While the aggregation is carried out at the aggregator node in each cluster, it is guaranteed that no individual node gets to know the sensitive private values of other nodes in the cluster. The intermediate aggregate value in each cluster is further aggregated along the routing tree as the data packets move to the sink node. The privacy goal of the scheme is two-fold. First, the privacy of data has to be guaranteed end-to-end. While only the sink could learn about the final aggregation result, each node will have information of its own data and does not have any information about the data of other nodes. Second, to reduce the communication overhead, the data from different source nodes have to be efficiently combined at the intermediate nodes (i.e. aggregation) along the path. Nevertheless, these intermediate nodes should not learn any information about the individual nodes' data. The authors of the CPDA scheme have presented performance results of the protocol to demonstrate the efficiency and security of the protocol. The CPDA protocol has become quite popular, and to the best of our knowledge, there has been no identified vulnerability of the protocol published in the literature so far.

In this paper, we first demonstrate a security vulnerability in the CPDA protocol and then proceed to show how the protocol may be made more efficient and secure. We also propose necessary modifications in the CPDA protocol to defend against the identified vulnerability.

The rest of this paper is organized as follows. Section 2 provides a brief background discussion on the CPDA scheme. In Section 3, we present a cryptanalysis on CPDA and demonstrate a security vulnerability of the scheme. In Section 4, we present some design modifications of the CPDA scheme. Section 4.1 presents an efficient way to compute the aggregation operation so as to make CPDA more efficient. Section 4.2 briefly discusses how the identified security vulnerability can be addressed. Section 5 presents a comparative analysis of the overhead of the original CPDA protocol and its proposed modified version. Section 5.1 provides a comparison of the communication overheads in the network, and Section 5.2 provides an analysis of the computational overheads in the sensor nodes. Section 6 concludes the paper while highlighting some future scope of work.

2 The CPDA Scheme [9] for Data Aggregation in WSN

The basic idea of CPDA is to introduce noise into the raw data sensed by the sensor nodes in a WSN, such that an aggregator can obtain accurate aggregated information but not individual sensor data [9]. This is similar to the data perturbation approach extensively used in privacy-preserving data mining. However, unlike in privacy-preserving data mining, where noises are independently generated (at random) leading to imprecise aggregated results, the noises in CPDA are carefully designed to leverage the cooperation between different sensor nodes, such that the precise aggregated values can be obtained by the aggregator.

The CPDA protocol classifies sensor nodes into two types: cluster leaders and cluster members. There is a one-to-many mapping between the cluster leaders and cluster members. The cluster leaders are responsible for aggregating data received from the cluster members. For security, the messages communicated between the cluster leaders and the cluster members are encrypted using different symmetric keys for each pair of nodes.

The details of the CPDA scheme are provided briefly in the following sub-sections.

2.1 The Network Model

The sensor network is modeled as a connected graph $G(V, E)$, where V represents the set of senor nodes and E represents the set of wireless links connecting the sensor nodes. The number of sensor nodes is taken as $|V| = N$.

A data aggregation function is taken that aggregates the individual sensor readings. CPDA scheme has focused on additive aggregation function, $f(t)= \sum_{i=1}^{N} d_i(t)$ where $d_i(t)$ is the individual sensor reading at time instant t for node i. For computation of the aggregate functions, the following requirements are to be satisfied: (i) privacy of the individual sensor data is to be protected, i.e., each node's data should be known to no other nodes except the node itself, (ii) the number of messages transmitted within the WSN for the purpose of data aggregation should be kept at a minimum, and (iii) the aggregation result should be as accurate as possible.

2.2 Key Distribution and Management

CPDA uses a random key distribution mechanism proposed in [6] for encrypting messages to prevent message eavesdropping attacks. The key distribution scheme has three phases: (i) key pre-distribution, (ii) shared-key discovery, and (iii) path-key establishment. These phases are described briefly as follows.

A large key-pool of K keys and their identities are first generated in the key pre-distribution phase. For each sensor nodes, k keys out of the total K keys are chosen. These k keys form a *key ring* for the sensor node.

During the key-discovery phase, each sensor node identifies which of its neighbors share a common key with itself by invoking and exchanging discovery messages. If a pair of neighbor nodes share a common key, then it is possible to establish a secure link between them.

In the path-key establishment phase, an end-to-end path-key is assigned to the pairs of neighboring nodes who do not share a common key but can be connected by two or more multi-hop secure links at the end of the shared-key discovery phase.

At the end of the key distribution phase, the probability that any pair of nodes possess at least one common key is given by (1).

$$p_{connect} = 1 - \frac{((K-k)!)^2}{(K-2k)!K!} \qquad (1)$$

If the probability that any other node can overhear the encrypted message by a given key is denoted as $p_{overhear}$, then $p_{overhear}$ is given by (2).

$$p_{overhear} = \frac{k}{K} \tag{2}$$

It has been shown in [9] that the above key distribution algorithm is efficient for communication in a large-scale sensor networks, and when a limited number of keys are available for encryption of the messages to prevent eavesdropping attacks.

2.3 Cluster-Based Private Data Aggregation (CPDA) Protocol

The CPDA scheme works in three phases: (i) cluster formation, (ii) computation of aggregate results in clusters, and (ii) cluster data aggregation. These phases are described below.

Cluster formation: Fig. 1 depicts the cluster formation process. A query server Q triggers a query by sending a *HELLO* message. When the *HELLO* message reaches a sensor node, it elects itself as a cluster leader with a pre-defined probability p_c. If the value of p_c is large, there will be more number of nodes which will elect themselves as cluster leaders. This will result in higher number of clusters in the network. On the other hand, smaller values of p_c will lead to less number of clusters due to fewer number of cluster leader nodes. Hence, the value of the parameter p_c can be suitably chosen to control the number of clusters in the network. If a node becomes a cluster leader, it forwards the *HELLO* message to its neighbors; otherwise, it waits for a threshold period of time to check whether any *HELLO* message arrives at it from any of its neighbors. If any *HELLO* message arrives at the node, it decides to join the cluster formed by its neighbor by

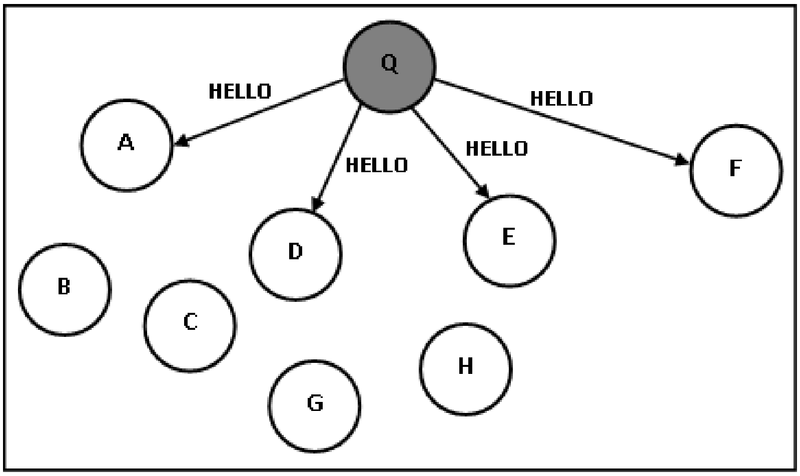

Fig. 1. Query server Q sends HELLO messages for initiating the cluster formation procedure to its neighbors A, D, E and F. The query server is shaded in the figure.

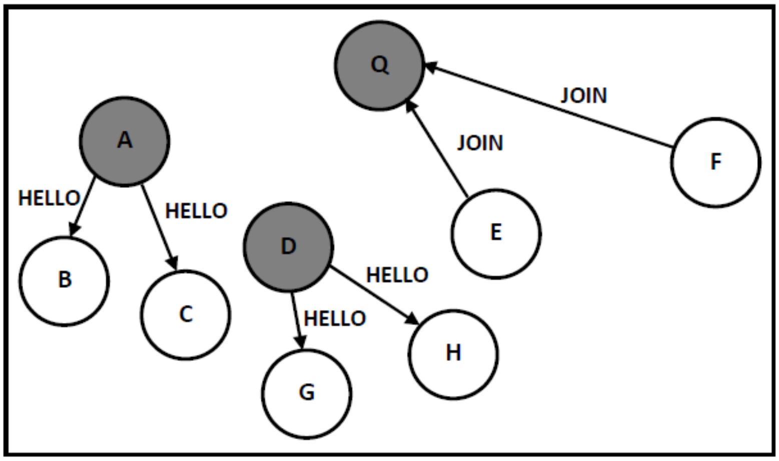

Fig. 2. A and D elect themselves as the cluster leaders randomly and in turn send *HELLO* messages to their neighbors. E and F join the cluster formed by Q. B and C join the cluster formed with A as the cluster leader, while G and H join the cluster with D as the cluster leader. All the cluster leaders and the query server are shaded.

broadcasting a *JOIN* message as shown in Fig 2. This process is repeated and multiple clusters are formed so that the entire WSN becomes a collection of a set of clusters.

Computation within clusters: In this phase, aggregation is done in each cluster. The computation is illustrated with the example of a simple case where a cluster contains three members: A, B, and C, where A is the assumed to be the cluster leader and the aggregator node, whereas B and C are the cluster member nodes. Let a, b, c represent the private data held by the nodes A, B, and C respectively. The goal of the aggregation scheme is to compute the sum of a, b and c without revealing the private values of the nodes.

As shown in Fig. 3, for the privacy-preserving additive aggregation function, the nodes A, B, and C are assumed to share three public non-zero distinct numbers, which are denoted as x, y, and z respectively. In addition, node A generates two random numbers r_1^A and r_2^A, which are known only to node A. Similarly, nodes B and C generate r_1^B, r_2^B and r_1^C, r_2^C respectively, which are private values of the nodes which have generated them.

Node A computes v_A^A, v_B^A, and v_C^A as shown in (3).

$$\begin{aligned} v_A^A &= a + r_1^A x + r_2^A x^2 \\ v_B^A &= a + r_1^A y + r_2^A y^2 \\ v_C^A &= a + r_1^A z + r_2^A z^2 \end{aligned} \tag{3}$$

Similarly, node B computes v_A^B, v_B^B, and v_C^B as in (4).

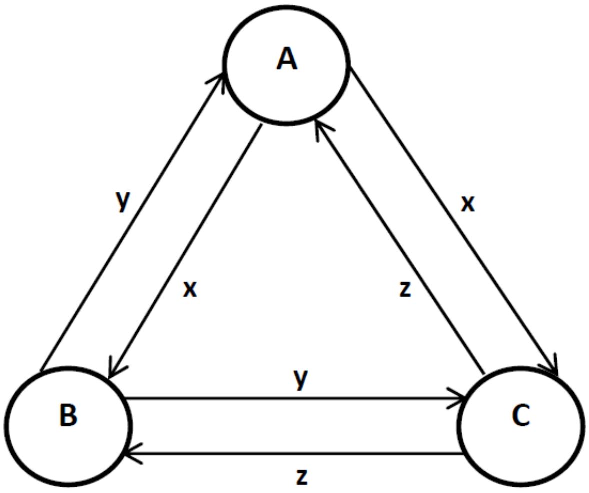

Fig. 3. Nodes A, B and C broadcast their distinct and non-zero public seeds x, y and z respectively

$$\begin{aligned} v_A^B &= b + r_1^B x + r_2^B x^2 \\ v_B^B &= b + r_1^B y + r_2^B y^2 \\ v_C^B &= b + r_1^B z + r_2^B z^2 \end{aligned} \tag{4}$$

Likewise, node C computes v_A^C, v_B^C, and v_C^C as in (5).

$$\begin{aligned} v_A^C &= c + r_1^C x + r_2^C x^2 \\ v_B^C &= c + r_1^C y + r_2^C y^2 \\ v_C^C &= c + r_1^C z + r_2^C z^2 \end{aligned} \tag{5}$$

Node A encrypts v_B^A and sends it to node B using the shared key between node A and node B. Node A also encrypts v_C^A and sends it to node C using the shared key between node A and node C. In the same manner, node B sends encrypted v_A^B to node A and v_C^B to node node C; node C sends encrypted v_A^C and v_B^C to node A and node B respectively. The exchanges of these encrypted messages is depicted in Fig. 4. On receiving v_A^B and v_A^C, node A computes the sum of v_A^A (already computed by node A), v_A^B and v_A^C. Now, node A computes F_A using (6).

$$F_A = v_A^A + v_A^B + v_A^C = (a + b + c) + r_1 x + r_2 x^2 \tag{6}$$

In (6), $r_1 = r_1^A + r_1^B + r_1^C$ and $r_2 = r_2^A + r_2^B + r_2^C$. Similarly, node B and node C compute F_B and F_C respectively, where F_B and F_C are given by (7) and (8) respectively.

$$F_B = v_B^A + v_B^B + v_B^C = (a + b + c) + r_1 y + r_2 y^2 \tag{7}$$

$$F_C = v_C^A + v_C^B + v_C^C = (a + b + c) + r_1 z + r_2 z^2 \tag{8}$$

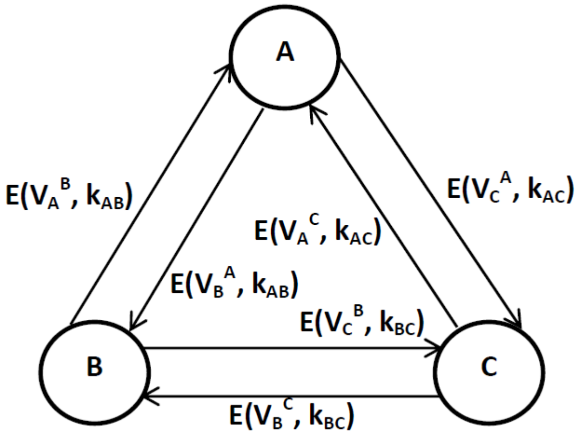

Fig. 4. Exchange of encrypted messages among nodes A, B and C using shared keys of the nodes

Node B and node C broadcast F_B and F_C to the cluster leader node A, so that node A has the knowledge of the values of F_A, F_B and F_C. From these values the cluster leader node A can compute the aggregated value $(a + b + c)$ as explained below.

The equations (6), (7), and (8) can be rewritten as in (9).

$$\mathbf{U} = \mathbf{G}^{-1}\mathbf{F} \tag{9}$$

$$\text{where, } \mathbf{G} = \begin{bmatrix} 1 \; x \; x^2 \\ 1 \; y \; y^2 \\ 1 \; z \; z^2 \end{bmatrix}, \mathbf{U} = \begin{bmatrix} a+b+c \\ r_1 \\ r_2 \end{bmatrix}, \text{ and } \mathbf{F} = \begin{bmatrix} F_A \; F_B \; F_C \end{bmatrix}^T.$$

Since x, y, z, F_A, F_B, and F_C are known to the cluster leader node A, it can compute the value of $(a + b + c)$ without having any knowledge of b and c.

In order to avoid eavesdropping attack by neighbor nodes, it is necessary to encrypt the values of v_B^A, v_C^A, v_A^B, v_C^B, v_A^C, and v_B^C. If node B overhears the value of v_C^A, then node B gets access to the values of v_C^A, v_B^A and F_A. Then node B can deduce $v_A^A = F_A - v_A^B - v_C^A$. Having the knowledge of v_A^A, node B can further obtain the value of a if x, v_A^A, v_A^B and v_A^C are known. However, if node A encrypts v_C^A and sends it to node C, then node B cannot get v_C^A. With the knowledge of v_B^A, F_A and x from node A, node B cannot deduce the value of a. If node B and node C collude and reveal node A's information (i.e., v_B^A and v_C^A), to each other, then node A's privacy will be compromised and its private value a will be revealed. In order to reduce the probability of such collusion attacks, the cluster size should be as large as possible, since in a cluster of size m, at least $(m$ - 1) nodes should collude in order to successfully launch the attack. Higher values of m will require larger number of colluding nodes thereby making the attack more difficult.

2.4 Cluster Data Aggregation

The CPDA scheme has been implemented on top of a protocol known as *Tiny Aggregation* (TAG) protocol [10]. Using the TAG protocol, each cluster leader node routes the sum of the values in the nodes in its cluster to the query server through a TAG routing tree whose root is situated at the server.

3 An Attack on the CPDA Scheme

In this section, we present an efficient attack on the CPDA aggregation scheme. The objective of the attack is to show the vulnerability of the CPDA scheme which can be suitably exploited by a malicious participating sensor node. The intention of the malicious node is to participate in the scheme in such a way that it can get access to the private values (i.e., a, b and c) of the participating sensor nodes. For describing the attack scenario, we use the same example cluster consisting of three sensor nodes A, B and C. Node A is the cluster leader whereas node B and node C are the cluster members. We distinguish two types of attacks: (i) attack by a malicious cluster leader (e.g., node A) and (ii) attack by a malicious cluster member (e.g., either node B or node C). These two cases are described in detail in the following sub-sections.

3.1 Privacy Attack by a Malicious Cluster Leader

Let us assume that the cluster leader node A is malicious. Node A chooses a very large value of x such that $x \gg y, z$. Since y and z are public values chosen by node B and node C which are broadcast in the network by node B and node C respectively, it is easy for node A to choose a suitable value for x.

Nodes A, B and C compute the values of v_A^A, v_B^A, v_C^A, v_A^B, v_B^B, v_C^B, v_A^C, v_B^C, and v_C^C using (3), (4) and (5) as described in Section 2.3. As per the CPDA scheme, node A receives $v_A^B = b + r_1^B x + r_2^B x^2$ from node B. Since x is very large compared to b and r_1^B, node A can derive the value of r_2^B using (10) where we consider integer division.

$$\frac{v_A^B}{x^2} = \frac{b}{x^2} + \frac{r_1^B}{x} + r_2^B = 0 + 0 + r_2^B = r_2^B \tag{10}$$

Using the value of r_2^B as derived in (10), and using $v_A^B = b + r_1^B x + r_2^B x^2$, node A can now compute the value of r_1^B by solving (11).

$$\frac{v_A^B - r_2^B x^2}{x} = \frac{b}{x} + r_1^B = 0 + r_1^B = r_1^B \tag{11}$$

In the same manner, node A derives the values of r_1^C and r_2^C from v_A^C received from node C. Since $r_1 = r_1^A + r_1^B + r_1^C$, and $r_2 = r_2^A + r_2^B + r_2^C$, as shown in (6), (7) and (8), node A can compute the values of r_1 and r_2 (r_1^B, r_2^B, r_1^C, and r_2^C are derived as shown above, and r_1^A and r_2^A were generated by node A).

At this stage, node A uses the values of F_B and F_C received from node B and node C respectively as shown in (7) and (8). Node A has now two linear simultaneous equations with two unknowns: b and c, the values of y and z being public. Solving (7) and (8) for b and c, the malicious cluster leader node A can get the access to the private information of nodes B and C, thereby launching a privacy attack on the CPDA scheme.

3.2 Privacy Attack by a Malicious Cluster Member

In this scenario, let us assume that the cluster member node B is malicious and it tries to access the private values of the cluster leader node A and the cluster member node C. Node B chooses a very large value of y so that $y \gg x$, z. Once the value of F_B is computed in (7), node B derives the value of r_2 and r_1 using (12) and (13).

$$\frac{F_B}{y^2} = \frac{(a+b+c)}{y^2} + \frac{r_1}{y} + r_2 = 0 + 0 + r_2 = r_2 \tag{12}$$

$$\frac{F_B - r_2 y^2}{y} = \frac{(a+b+c)}{y} + r_1 = 0 + r_1 = r_1 \tag{13}$$

As per the CPDA scheme, node B receives $v_B^C = c + r_1^C y + r_2^C y^2$ from node C. Since the magnitude of y is very large compared to c, r_1^C and r_2^C, it is easy for node B to derive the values of r_2^C and r_1^C using (14) and (15).

$$\frac{v_B^C}{y^2} = \frac{c}{y^2} + \frac{r_1^C}{y} + r_2^C = 0 + 0 + r_2^C = r_2^C \tag{14}$$

$$\frac{v_B^C - r_2^C y^2}{y} = \frac{c}{y} + r_1^C = 0 + r_1^C = r_1^C \tag{15}$$

Using (12), (13), (14) and (15) node B can compute $r_1^A = r_1 - r_1^B - r_1^C$, and $r_2^A = r_2 - r_2^B - r_2^C$. Now, node B can compute the value of a using $v_A^B = a + r_1^A y + r_2^A y^2$ (received from node A), in which the values of all the variables are known except that of a. In a similar fashion, node B derives the value of c using $v_B^C = c + r_1^C y + r_2^C y^2$ (received from C).

Since the private values of the nodes A and C are now known to node B, the privacy attack launched by participating cluster member node B is successful on the CPDA aggregation scheme.

4 Modifications of CPDA

In this section, we present two modifications of CPDA scheme: one towards making the protocol more efficient and the other for making it more secure.

4.1 Modification of CPDA for Enhanced Efficiency

In this section, a modification is proposed for the CPDA protocol for achieving enhanced efficiency in its operation. The modification is based on suitable choice for the value of x (the public seed) done by the aggregator node A.

Let us assume that the node A chooses a large value of x such that the following conditions in (16) and (17) are satisfied.

$$r_2 x^2 \gg r_1 x \tag{16}$$

$$r_1 x \gg (a + b + c) \tag{17}$$

where, $r_1 = r_1^A + r_1^B + r_1^C$ and $r_2 = r_2^A + r_2^B + r_2^C$. Now, node A has computed the value of F_A as shown in (6). In order to efficiently compute the value of (a + b + c), node A divides the value of F_A by x^2 as shown in (18).

$$\frac{F_A}{x^2} = \frac{(a+b+c)}{x^2} + \frac{r_1 x}{x^2} + r_2 = 0 + 0 + r_2 = r_2 \tag{18}$$

Using (18), node A derives the value of r_2. Once the value of r_2 is deduced, node A attempts to compute the value of r_1 using (19) and (20).

$$F_A - r_2 x^2 = (a + b + c) + r_1 x \tag{19}$$

$$r_1 = \frac{(F_A - r_2 x^2)}{x} - \frac{(a+b+c)}{x} = \frac{(F_A - r_2 x^2)}{x} - 0 = \frac{(F_A - r_2 x^2)}{x} \tag{20}$$

Since, the values of F_A, r_2 and x are all known to node A, it can compute the value of r_1 using (20). Once the values of r_1 and r_2 are computed by node A, it can compute the value of (a + b + c) using (6). Since the computation of the sum (a + b + c) by node A involves two division operations (involving integers) only (as done in (18) and (20)), the modified CPDA scheme will be light-weight and hence much more energy- and time-efficient as compared to the original CPDA scheme. The original CPDA scheme involved additional computations of the values of F_B and F_C, as well as an expensive matrix inversion operation as described in Section 2.3.

4.2 Modification of CPDA for Resisting the Attack

In this section, we discuss the modifications required on the existing CPDA scheme so that a malicious participant node cannot launch the attack described in Section 3.

It may be noted that, the vulnerability of the CPDA scheme lies essentially in the unrestricted freedom delegated on the participating nodes for generating their public seed values. For example, nodes A, B and C have no restrictions on their choice for values of x, y and z respectively while they generate these values. A malicious attacker can exploit this freedom to generate an arbitrarily large public seed value, and can thereby launch an attack as discussed in Section 3.

In order to prevent such an attack, the CPDA protocol needs to be modified. In this modified version, the nodes in a cluster make a check on the generated public seed values so that it is not possible for a malicious participant to generate any arbitrarily large seed value. For a cluster with three nodes, such a constraint may be imposed by the requirement that the sum of any two public seeds must be greater than the third seed. In other words: $x + y > z$, $z + x > y$, and $y + z > x$. If these constraints are satisfied by the generated values of x, y and z, it will be impossible for any node to launch the attack and get access to the private values of the other participating nodes.

However, even if the above restrictions on the values of x, y and z are imposed, the nodes should also be careful in choosing the values for their secret random number pairs. If two nodes happen to choose very large values for their random numbers compared to those chosen by the third node, then it will be possible for the third node to get access to the private values of the other two nodes. For example, let us assume that nodes A and C have chosen the values of r_1^A, r_2^A and r_1^C, r_2^C such that they are all much larger than r_1^B and r_2^B - the private random number pair chosen by node B. It will be possible for node B to derive the values of a and c : the private values of nodes A and C respectively. This is explained in the following.

Node B receives $v_B^A = a + r_1^A y + r_2^A y^2$ from node A and computes the values of r_1^A and r_2^A using (21) and (22).

$$\frac{v_B^A}{y^2} = \frac{a}{y^2} + \frac{r_1^A}{y} + r_2^A = 0 + 0 + r_2^A = r_2^A \tag{21}$$

$$\frac{v_B^A - r_2^A y^2}{y} = \frac{a}{y} + r_1^A = 0 + r_1^A = r_1^A \tag{22}$$

In a similar fashion, node B derives the values of r_1^C and r_2^C from v_B^C received from node C. Now, node B computes $r_1 = r_1^A + r_1^B + r_1^C$ and $r_2 = r_2^A + r_2^B + r_2^C$ since it has access to the values of all these variables. In the original CPDA scheme in [9], the values of F_B and F_C are broadcast by nodes B and C in unencrypted from. Hence, node B has access to both these values. Using (7) and (8), node B can compute the values of a and c, since these are the only unknown variables in the two linear simultaneously equations.

In order to defend against the above vulnerability, the CPDA protocol needs further modification. In this modified version, after the values v_A^A, v_A^B, and v_A^C are generated and shared by nodes A, B and C respectively, the nodes check whether the following constraints are satisfied: $v_A^A + v_A^B > v_A^C$, $v_A^B + v_A^C > v_A^A$, and $v_A^C + v_A^A > v_A^B$. The nodes proceed for further execution of the algorithm only if the above three inequalities are satisfied. If all three inequalities are not satisfied, there will be a possibility that the random numbers generated by one node is much larger than those generated by other nodes - a scenario which indicates a possible attack by a malicious node.

5 Performance Analysis

In this section, we will briefly present a comparative analysis of the overheads of the CPDA protocol and its proposed modified versions presented in Section 4.1 and Section 4.2. Our analysis is based on two categories of overheads: (i) overhead due to message communication in the network and (ii) computational overhead at the sensor nodes.

5.1 Communication Overhead

We compare communication overheads of three protocols - the *tiny aggregation protocol* (TAG), the original CPDA protocol and the proposed modified CPDA protocols. In TAG, each sensor node needs to send 2 messages for the data aggregation protocol to work. One *Hello* message communication from each sensor node is required for forming the aggregation tree, and one message is needed for data aggregation. However, this protocol only performs data aggregation and does not ensure any privacy for the sensor data. In the original CPDA protocol, each cluster leader node sends 4 messages and each cluster member node sends 3 messages for ensuring that the aggergation protocol works in a privacy-preserving manner. In the example cluster shown in Fig. 3, the 4 messages sent by the cluster leader node A are: one *Hello* message for forming the cluster, one message for communicating the public seed x, one message for communicating v_B^A and v_C^A to cluster member nodes B and C respectively, and one message for sending the aggregate result from the cluster. Similarly, the 3 messages sent by the cluster member node B are: one message for communicating its public seed y, one message for communicating v_A^B and v_C^B to cluster leader node A and cluster member node C respectively, and one message for communicating the intermediate result F_B to the cluster leader node A.

In contrast to the original CPDA protocol, the modified CPDA protocol in Section 4.1 involves 3 message communications from the cluster leader node and 2 message communications from each cluster member node. The 3 messages sent by the cluster leader node A are: one *Hello* message for forming the cluster, one message for brodcasting its public seed x, and one message for sending the final aggregate result. It may be noted that in this protocol, the cluster leader node A need not send v_B^A and v_C^A to the cluster member nodes B and C respectively. Each cluster member node needs to send 2 messages. For example, the cluster member node B needs to broadcast its public seed y, and also needs to send v_A^B to the cluster leader node A. Unlike in the original CPDA protocol, the cluster member node B does not send F_B to the cluster leader. Similarly, the cluster member node C does not send F_C to the cluster leader node A. In a cluster consisting of three members, the original CPDA protocol would involve 10 messages (4 messages from the cluster leader and 3 messages from each cluster member). The modified CPDA protocol presented in Section 4.1, on the other hand, would involve 7 messages (3 messages from the cluster leader and 2 messages from each cluster member) in a cluster of three nodes. Therefore, in a cluster of three nodes, the modified CPDA protocol presented in Section 4.1 will involve 3 less message communications. Since in a large-scale WSN the number of clusters will be quite high, there will be an appreciable reduction in the communication overhead in the modified CPDA protocol presented in Section 4.1.

The secure version of the modified CPDA protocol presented in Section 4.2 involves the same communication overhead as the original CPDA protocol. However, if any node chooses abnormally higher values for its public seed or its private random numbers, the secure version of the modified CPDA protocol will involve 2 extra messages from each of the participating sensor nodes. Therefore, in a cluster of three nodes, the secure version of the modified CPDA protocol will involve 6 extra messages in the worst case scenario when compared with the original CPDA protocol.

If p_c is the probability of a sensor node electing itself as a cluster leader, the average number of messages sent by a sensor node in the original CPDA protocol is: $4p_c + 3(1 - p_c) = 3 + p_c$. Thus, the message overhead in the original CPDA is less than twice as that in TAG. However, in the modified CPDA protocol presented in Section 4.1, the average number of messages communicated by a sensor node is: $3p_c + 2(1 - p_c) = 2 + p_c$. As mentioned in Section 2.3, in order to prevent collusion attack by sensor nodes, the cluster size in the CPDA protocol should be as large as possible. This implies that the value of p_c should be small. Since the value of p_c is small, it is clear that the message overhead in the modified CPDA protocol presented in Section 4.1 is almost the same as that in TAG and it is much less (one message less for each sensor node) than that of the original CPDA protocol. In the secure version of the protocol in Section 4.2, the communication overhead, in the average case, will be the same as in the original CPDA protocol. However, in the worst case, the number of messages sent by a sensor node in this protocol will be: $6p_c + 5(1 - p_c) = 5 + p_c$. This is 2.5 times the average communication overhead in the TAG protocol and 1.67 times the average communication overhead in the original CPDA protocol. The secure protocol, therefore, will involve 67% more overhead in the worst case scenario (where a malicious participant sensor node chooses abnormally higher values for its public seed as well as for its private random numbers).

5.2 Computational Overhead

In this section, we present a comparative analysis of the computational overheads incurred by the sensor nodes in the original CPDA protocol and in the proposed efficient version of the protocol.

Computational overhead of the original CPDA protocol: The overhead of the CPDA protocol can be broadly classified into four categories: (i) computation of the parameters, (ii) computation for encrypting messages, (iii) computation of the intermediate results, and (iv) computation of the final aggregate result at the cluster leader node. The details of these computations are presented below:

(i) *Computation of the parameters at the sensor nodes*: Each sensor node in a three member cluster computes three parameters. For example, the cluster leader node A computes v_A^A, v_B^A and v_C^A. Similarly, the cluster member node B computes v_A^B, v_B^B and v_C^B. We first compute the overhead due these computations.

Since $v_A^A = a + r_1^A x + r_2^A x^2$, for computation of v_A^A, node A needs to perform 2 addition, 2 multiplication and 1 exponentiation operations. Hence, for computing v_A^A, v_B^A, v_C^A, node A needs to perform 6 addition, 6 multiplication and 3 exponentiation operations. Therefore, in a cluster consisting of three members, for computation of all parameters, the original CPDA protocol requires 18 addition, 18 multiplication and 9 exponentiation operations.

(ii) *Computations for encrypting messages*: Some of the messages in the CPDA protocol need to be communicated in encrypted form. The encryption operation involves computaional overhead. For example, node A needs to encrypt v_B^A and v_C^A before sending them to nodes B and C respectively. Therefore, 2 encryption operations are required at node A. For a cluster consisting of three members, the CPDA protocol will need 6 encryption operations.

(iii) *Computations of intermediate results*: The nodes A, B, and C need to compute the intermediate values F_A, F_B and F_C respectively. Since $F_A = v_A^A + v_A^B + v_A^C = (a + b + c) + r_1 x + r_2 x^2$ and $r_1 = r_1^A + r_1^B + r_1^C$, and $r_2 = r_2^A + r_2^B + r_2^C$, for computing F_A, node A will need to perform 4 addition operations. Therefore, for a cluster of three members, 12 addition operations will be needed.

(iv) *Aggregate computation at the cluster leader*: For computing the final aggregated result in a privacy-preserving way, the cluster leader node A needs to perform one matrix inversion operation and one matrix multiplication operation.

The summary of various operations in the original CPDA protocol are presented in Table 1.

Computational overhead of the modified CPDA protocol: The overhead of the efficient version of the CPDA protocol presented in Section 4.1 are due to: (i) computation of the parameters at the sensor nodes, (ii) computation of the intermediate result at the cluster leader node, and (iii) computation of the aggregated result at the cluster leader node. The details of these computations are presented below.

Table 1. Operations in the CPDA Protocol

Operation Type	**No. of Operations**
Addition	30
Multiplication	18
Exponentiation	3
Encryption	6
Matrix Multiplication	1
Matrix Inversion	1

(i) *Computation of the parameters at the sensor nodes*: In the modified version of the CPDA protoocl, the nodes A, B and C need to only compute v_A^A, v_A^B, and v_A^C respectively. As shown earlier, each parameter computation involves 2 addition, 2 multiplication and 1 exponentiation operations. Therefore, in total, 6 addition, 6 multiplication, and 3 exponentiation operations will be needed.

(ii) *Computations for encrypting messages*: The nodes B and C will need to encrypt the messages v_A^B and v_A^C respectively before sending them to the cluster leader node A. Therefore, 2 encryption operations will be required.

(iii) *Computation of intermediate result*: The cluster leader node A will only compute F_A in the modified CPDA. The cluster member nodes B and C need not perform any computations here. As discussed earlier, computation of F_A needs 4 addition operations.

(iv) *Aggregate computation at the cluster leader*: For computation of the final result at the cluster leader node, 2 integer division and 2 subtraction operations will be required.

The summary of various operations in the modified CPDA protocol are presented in Table 2.

Table 2. Operations in the Proposed Modified CPDA Protocol

Operation Type	**No. of Operations**
Addition	10
Subtraction	2
Multiplication	6
Division	2
Exponentiation	3
Encryption	2

It is clearly evident from Table 1 and Table 2 that the modified version of the CPDA protocol involves much less computational overhead than the original version of the protocol.

6 Conclusion

Innetwork data aggregation in WSNs is a technique that combines partial results at the intermediate nodes en route to the base station (i.e. the node issuing the query), thereby reducing the communication overhead and optimizing the bandwidth utilization in the wireless links. However, this technique raises privacy issues of the sensor nodes which need to share their data with the aggregator node. In applications such as health care and military surveillance where the sensitivity of the private data of the sensors is very high, the aggregation has to be carried out in a privacy-preserving way, so that the sensitive data are not revealed to the aggregator. A very popular scheme for this purpose exists in the literature which is known as CPDA. Although CPDA is in literature for quite some time now, no vulnerability of the protocol has been identified so far. In this paper, we have first demonstrated a security vulnerability in the CPDA protocol, wherein a malicious sensor node can exploit the protocol is such a way that it gets access to the private values of its neighbors while participating in data aggregation process. A suitable modification of the CPDA protocol is further proposed so as to plug the identified vulnerability and also to make the protocol computationally more efficient. We have also made an analysis of the communication and computational overhead in the original CPDA protocol and the proposed modified version of the CPDA protocol. It has been found from the analysis that the modified version of the protocol involves appreciably less message communication overhead in the network and computational load on the sensor nodes.

It may be noted that over the past few years, several schemes have been proposed in the literature for privacy preserving data aggregation in WSNs. A very popular and elegant approach in this direction is *homomorphic encryption* [7]. Westhoff et al. have proposed additive privacy homomorphic functions that allow for end-to-end encryption between the sensors and the sink node and simultaneously enable aggregators to apply aggregation functions directly over the ciphertexts [13]. This has the advantage of eliminating the need for intermediate aggregators to carry out decryption and encryption operations on the sensitive data. Armknecht et al. have presented a symmetric encryption scheme for sensor data aggregation that is homomorphic both for data and the keys [3]. This is called *bi-homomorphic encryption*, which is also essentially an additive homomorphic function. Castellucia et al. have proposed an approach that combines inexpensive encryption techniques with simple aggregation methods to achieve efficient aggregation of encrypted data in WSNs [4]. The method relies on end-to-end encryption of data and hop-by-hop authentication of nodes. Privacy is achieved by using additive homomorphic functions. A very simple approach for privacy-preserving multi-party computation has been discussed by Chaum [5]. The protocol is known as *Dining Cryptographers Problem* which describes the way a channel is created so that it is difficult to trace (i.e. identify) the sender of any message through that channel.

The approaches based on privacy homomorphic functions are more elegant than CPDA for the purpose of carrying out sensor data aggregation in a privacy preserving way. However, they involve large computational overhead due to complexities involved in computing the homomorphic encryption functions and the associated key management related issues. The primary objective of our work in this paper is to demonstrate a security vulnerability of the CPDA protocol. We plan to evaluate the performance of our modified CPDA protocol as a future scope of work, and make a comparative analysis of the protocol with the existing privacy homomorphism-based approaches for sensor data aggregation.

References

1. Acharya, M., Girao, J., Westhoff, D.: Secure Comparison of Encrypted Data in Wireless Sensor Networks. In Proc. of the 3rd International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WIOPT), Washington, DC, USA, pp. 47–53 (2005)
2. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless Sensor Networks: A Survey. Computer Networks 38(4), 393–422 (2002)
3. Armknecht, F., Westhoff, D., Girao, J., Hessler, A.: A Lifetime-Optimized End-to-End Encryption Scheme for Sensor Networks Allowing In-Network Processing. Computer Communications 31(4), 734–749 (2008)
4. Castelluccia, C., Chan, A.C-F., Mykletun, E., Tsudik, G.: Efficient and Provably Secure Aggregation of Encrypted Data in Wireless Sensor Networks. ACM Transactions on Sensor Networks 5(3) (2009)
5. Chaum, D.: The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. Journal of Cryptology 1(1), 65–75 (1988)
6. Eschenauer, L., Gligor, V.D.: A Key-Management Scheme for Distributed Sensor Networks. In: Proc. of the 9th ACM Conference on Computing and Communications Security, pp. 41–47 (2002)
7. Fontaine, C., Galand, F.: A Survey of Homomorphic Encryption for Nonspecialists. EURASIP Journal on Information Security 2007, Article ID 13801 (2007)
8. Girao, J., Westhoff, D., Schneider, M.: CDA: Concealed Data Aggregation for Reverse Multicast Traffic in Wireless Sensor Networks. In: Proc. of the 40th IEEE Conference on Communications (IEEE ICC), vol. 5, pp. 3044–3049 (2005)
9. He, W., Liu, X., Nguyen, H., Nahrstedt, K., Abdelzaher, T.: PDA: Privacy-Preserving Data Aggregation in Wireless Sensor Networks. In Proc. of the 26th IEEE International Conference on Computer Communications (IEEE INFOCOM), pp. 2045–2053 (2007)
10. Madden, S., Franklin, M.J., Hellerstein, J.M., Hong, W.: TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks. In Proc. of the 5th Symposium on Operating Systems Design and Implementation (OSDI), vol. 36 (2002)
11. Peter, S., Westhoff, D., Castelluccia, C.: A Survey on the Encryption of Convergecast Traffic with In-Network Processing. IEEE Transactions on Dependable and Secure Computing 7(1), 20–34 (2010)
12. Sen, J.: A Survey on Wireless Sensor Network Security. International Journal of Communication Networks and Information Security (IJCNIS) 1(2), 59–82 (2009)
13. Westhoff, D., Girao, J., Acharya, M.: Concealed Data Aggregation for Reverse Multicast Traffic in Sensor Networks: Encryption, Key Distribution, and Routing Adaptation. IEEE Transactions on Mobile Computing 5(10), 1417–1431 (2006)