Cluster Based Privacy Preserving Data Aggregation Algorithm for Wireless Sensor Networks

1 Xianyou Zhu, 2, 3 Lei Wang

1 Department of Computer Science, Hengyang Normal University, 421008, P. R. China
2 College of Information Engineering, Xiangtan University, 411105, P. R. China
3 Key Laboratory of Intelligent Computing & Information Processing (Xiangtan University), 411105, P. R. China

1 Tel.: +86 734 8484918, fax: +86 734 3456048
1 E-mail: 24754887@qq.com

Received: 23 February 2014 /Accepted: 31 July 2014 /Published: 31 August 2014

Abstract: Providing efficient data privacy and data aggregation simultaneously is a challenging work in wireless sensor networks. In this paper, we propose a novel queue based privacy-preserving data aggregation scheme for additive aggregation function. In the scheme, sensor nodes are divided into clusters in a distributed way first, and then, in each cluster, cluster members will form a queue and adopt a new queue based privacy-preserving scheme to send data for aggregation upon the cluster head’s request. We evaluate our scheme in terms of communication/storage/computational overheads, capacity of privacy - preservation, and data aggregation accuracy. The results show that our scheme is efficient and reasonable.

Keywords: Privacy preservation, Data aggregation, Wireless sensor network.

1. Introduction

In certain wireless sensor network (WSN) applications, privacy-preserving aggregation of data is required during transmission from the source sensor nodes to the sink node [1-7], and then, we need a way to collect the aggregated sensor readings while preserving data privacy of individual sensors at the same time [8-11]. Recently, researchers have proposed many efficient methods to bridge the gap between collaborative data collection and data privacy in WSN. For example, He et al. [12] proposed a distributed access control protocol with privacy support in WSN, which allows the network to authorize and grant user access privileges for data access within the network and can achieve privacy-preserving access control. Zhang et al. [13] proposed a perturbed histogram-based aggregation method to preserve privacy for queries targeted at special sensor data or sensor data distribution, which utilizes perturbation technique to hide the actual individual readings and the actual aggregate results sent by sensor nodes. He et al. [14] proposed a data aggregation protocol called integrity-protecting private data aggregation, in which two disjoint aggregation trees are constructed to achieve integrity and slicing/assembling techniques are adopted to achieve data privacy. Ukil et al. [15] considered a scenario in which two or more parties that own confidential data need to share only for aggregation purpose to a third party, without revealing the content of the data, and they proposed a scheme based on the secure key management scheme and randomized data perturbation technique to provide privacy preservation.
In this paper, we propose a privacy-preserving data aggregation scheme named Cluster-based Privacy - Preserving Data Protocol (CP²DA) for additive aggregation functions in WSN. In the CP²DA scheme, sensor nodes are divided into clusters in a distributed way first, and then, within each cluster, a node with maximum residual energy will be elected as the Cluster Head, and other sensor nodes will become the Cluster Member. In addition, for any two nodes \( A \) and \( B \) in the same cluster, a unique pair-wise key \( K_{A,B} \) will be established between them either. After that, for a cluster \( C \) with a cluster head \( CH \) and \( n \) cluster members \( CM_1, CM_2, ..., CM_n \), when the cluster head needs to collect data from \( CM_1, CM_2, ..., CM_n \), for each cluster member \( CM_i, i \in \{1, 2, ..., n\} \), it will slice its private data \( d_i \) into \( n \) different pieces \( d_{i1}, d_{i2}, ..., d_{in} \) randomly first, and then, for each piece \( d_{ij}, j \in \{1, 2, ..., n\} \), \( CM_i \) will add two different selected random numbers to \( d_{ij} \), and let the result be \( D_{ij} \), then, \( CM_i \) will keep \( D_{ij} \) for itself, and send \( D_{ij}, j \in \{1, 2, ..., i-1, i+1, ..., n\} \), to the cluster member \( CM_j \), after encrypting \( D_{ij} \) with the pair-wise key between itself and \( CM_j \). Finally, for each cluster member \( CM_i, i \in \{1, 2, ..., n\} \), after receiving these encrypted \( D_{i1}, D_{i2}, ..., D_{i,i-1}, D_{i,i+1}, ..., D_{in} \) from all other \( n \) cluster members \( CM_1, CM_2, ..., CM_{i-1}, CM_{i+1}, ..., CM_n \), respectively, it will decrypt them, and then, it will compute the sum of \( D_{ij} \) and encrypt it with the air-wise key between itself and the cluster head \( CH \), and then, send the encrypted data to \( CH \). For the head \( CH \), after receiving these encrypted data from all its \( n \) cluster members \( CM_1, CM_2, ..., CM_n \), respectively, it can decrypt them and compute out the sum of original data \( \sum_{i=1}^{n} d_i \) without knowing the detail of \( d_i \).

Simulation results show that adopting the CP²DA scheme proposed by us, data privacy can be well preserved while data aggregation is carrying out effectively, adopting the CP²DA scheme proposed by us, data privacy can be well preserved while data aggregation is carrying out effectively.

### 2. System Models

#### 2.1. Network Model

In this paper, a WSN is modeled as a connected graph \( G(V,E) \), where sensor nodes are represented as the set of vertices \( V \) and wireless links as the set of edges \( E \). In addition, we suppose there is one Sink node and \( N \) sensor nodes deployed in the WSN, and all sensor nodes have the same valid communication radius \( R \). For any two sensor nodes in the WSN, if the distance between them is less than or equals to \( R \), then we say there exists a wireless link between them and the two nodes are called as neighboring nodes.

After deployment, suppose the coordinates of the Sink node is \( (X_0, Y_0) \), and for any sensor node \( P_i, i \in \{1, 2, ..., N\} \), in the WSN, suppose its coordinates is \( (X_i, Y_i) \), then, we denote that the node \( N_i \) belongs to the cluster \( (\alpha, \beta) \), if and only if the following Formula (1) and Formula (2) hold.

\[
\alpha = \left[ \frac{(X_i - X_0)}{\left(\frac{R}{\sqrt{2}}\right)} \right], \quad (1)
\]

\[
\beta = \left[ \frac{(Y_i - Y_0)}{\left(\frac{R}{\sqrt{2}}\right)} \right], \quad (2)
\]

where the operator ‘\( \left[ x \right] \)’ returns the first integer that is bigger than or equals to \( x \).

Obviously, according to the above Formula (1) and Formula (2), all sensor nodes deployed in the sensing field can be divided into different clusters in a distributed way as follows.

Step 1: The Sink node obtains its coordinates \( (X_0, Y_0) \), and broadcasts a Declaration Message including its coordinates \( (X_0, Y_0) \) and node ID (\( P_i \)) to all its neighboring nodes.

Step 2: Except for the Sink node, for any sensor node \( P_i, i \in \{1, 2, ..., N\} \), when it received a Declaration Message, then,

Step 2.1: If it is the first time that \( P_i \) received this message, \( P_i \) will record \( (X_0, Y_0) \) in its Information Table, and then, it will broadcast the Declaration Message including the coordinates \( (X_0, Y_0) \) of the Sink node and its own node ID (\( P_i \)) to all its neighboring nodes.

Step 2.2: Otherwise, \( P_i \) will ignore the Declaration Message.

Step 2.3: After recording \( (X_0, Y_0) \) in its Information Table, then, \( P_i \) obtains its coordinates \( (X_i, Y_i) \). Hence, according to the Formula (1) and Formula (2), \( P_i \) can obtain its Cluster Number \( (\alpha, \beta) \) based on \( (X_0, Y_0) \) and \( (X_i, Y_i) \).

Step 2.4: After receiving the Declaration Messages forwarded by all its neighboring nodes, \( P_i \) will know the node IDs of all its neighboring nodes.

Observing the above Fig. 1, it is obvious that the whole sensing field can be divided into 12 clusters such as \((-1,0), (0,0), (1,0), (2,0), (-1,1), (0,1), (1,1), (2,1), (-1,2), (0,2), (1,2), (2,2)\) by all the sensor nodes in a distributed way.

![Fig. 1. Clusters forming example.](image-url)
Based on the above Formula (1) and Formula (2), it is easy to prove the following theorems hold.

**Theorem 1:** For any two sensor nodes $P_i$ and $P_j$, if they belong to the same cluster, then $P_i$ and $P_j$ are neighboring nodes.

**Proof.** Suppose the coordinates of the Sink node is $(X_0, Y_0)$, and the coordinates of $P_i$ and $P_j$ are $(X_i, Y_i)$ and $(X_j, Y_j)$ respectively, then from the Formula (1) and Formula (2), it is easy to know that the following four formulas hold.

\[
\frac{(\alpha - 1)R}{\sqrt{2}} \leq X_i - X_0 \leq \frac{\alpha R}{\sqrt{2}}
\]

\[
\frac{(\alpha - 1)R}{\sqrt{2}} \leq X_j - X_0 \leq \frac{\alpha R}{\sqrt{2}}
\]

\[
\frac{(\beta - 1)R}{\sqrt{2}} \leq Y_i - Y_0 \leq \frac{\beta R}{\sqrt{2}}
\]

\[
\frac{(\beta - 1)R}{\sqrt{2}} \leq Y_j - Y_0 \leq \frac{\beta R}{\sqrt{2}}
\]

From the Formulas (3), (4), (5), (6), it is easy to prove that the following two formulas hold.

\[
|X_i - X_j| \leq \frac{R}{\sqrt{2}}
\]

\[
|Y_i - Y_j| \leq \frac{R}{\sqrt{2}}
\]

Hence, the distance between $P_i$ and $P_j$ satisfies $\sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2} \leq R$, which means that $P_i$ and $P_j$ are neighboring nodes.

### 2.2. Security Key Establishment Model

After all sensor nodes deployed in the sensing field are divided into different clusters according to the scheme introduced in section 3.1, within each cluster $(\alpha, \beta)$, suppose there are $n+1$ different sensor nodes in $(\alpha, \beta)$, these $n+1$ sensor nodes will elect a cluster head and establish pair-wise keys between each other as follows.

Step 1: For any sensor node $P_i \in \{\alpha, \beta\}, i \in \{1, 2, ..., n+1\}$, it will broadcast a Hello Message including its Node ID, Residual Energy and Cluster Number $(\alpha, \beta)$ to all its neighboring nodes.

Step 2: After receiving the Hello Messages from all its neighboring nodes, $P_i$ records the information of Node ID, Residual Energy and Cluster Number of all its neighboring nodes in its Information Table, and then, it can distinguish these $n$ neighboring nodes with Cluster Number $(\alpha, \beta)$ from all its neighboring nodes. Suppose $P_i$ has totally $n+m$ different neighboring nodes, and the front $n$ different neighboring nodes $P_2$, $P_3$, ..., $P_{n+1}$ have the same Cluster Number $(\alpha, \beta)$, and the latter $m$ different neighboring nodes $P_{n+2}$, $P_{n+3}$, ..., $P_{n+m+1}$ have Cluster Number different from $P_i$, then

Step 2.1: Let the maximum Residual Energy among $\{P_1, P_2, ..., P_{n+1}\}$ be $E_{\text{max}}$, the maximum Node ID among $\{P_1, P_2, ..., P_{n+1}\}$ be $ID_{\text{max}}$, if the Residual Energy of $P_i$ equals to $E_{\text{max}}$ and the Node ID of $P_i$ equals to $ID_{\text{max}}$, then, $P_i$ elects itself as the cluster head of the cluster $(\alpha, \beta)$.

Step 2.2: $P_i$ generates $n+m$ random numbers $R_{0,0}$, ..., $R_{n,0}$, $R_{n,n+1}$, ..., $R_{n,n+m}$. To each of its neighboring node $P_j \in \{P_1, P_2, P_{n+1}, ..., P_{n+2}, ..., P_{n+m+1}\}$, $P_i$ generates the pair-wise key $K_{ij}$ between itself and $P_j$ as:

\[
K_{ij} = \text{Hash}(R_{i,j} + R_{j,i})
\]

### 2.3. Data Aggregation Model

A data aggregation function is defined as $y(t) = f(d_1(t), d_2(t), ..., d_K(t))$, where $d_i(t)$ is the individual sensor reading at time $t$ for node $P_i$. Typical functions of $f$ include sum, average, min, max and count etc. Without losing generality, in this paper, we focus on additive aggregation functions, that is,

\[
f(d_1(t), d_2(t), ..., d_K(t)) = \sum_{i=1}^{K} d_i(t)
\]

Obviously, if $d_i(t)$ $(i=1, 2, ..., K)$ is given, the computation of $y(t)$ at a query server (data sink) is trivial. However, due to the heavy data traffic in sensor networks, bandwidth constraints on wireless links, and large power consumption of packet transmission, data aggregation techniques are required to save resources and power.

### 3. Cluster based Privacy-Preserving Data Aggregation Scheme

For any cluster $(\alpha, \beta)$, suppose there are total $n+1$ sensor nodes $\{P_1, P_2, ..., P_{n+1}\}$ in the cluster $(\alpha, \beta)$, and $P_i$ is the cluster head, and $\{P_2, P_3, ..., P_{n+1}\}$ are cluster members, and in addition, suppose $d_i(t)$ is the individual sensor reading at time $t$ for node $P_i$ $(i=2, 3, ..., n+1)$, then, when cluster head $P_1$ wants to collect the sum of readings $D(t) = \sum_{i=2}^{n+1} d_i(t)$ from its cluster members $\{P_2, P_3, ..., P_{n+1}\}$ without knowing the detail of $d_i(t)$ $(i=2, 3, ..., n+1)$, it can adopt the following steps:
Step 1: For each cluster member $P_i$ ($i=2, 3, \ldots, n+1$), it slices its private data $d(t)$ into $n$ different pieces $d_2(t), d_3(t), \ldots, d_{n+1}(t)$ randomly, where

\[
d(t) = \sum_{j=2}^{n+1} d_j(t)
\]  

(11)

And then, the data $D(t)$ desired by the cluster head $P_i$ can be represented as follows:

\[
D(t) = \sum_{i=2}^{n+1} d_j(t) = \sum_{i=2}^{n+1} d_j(t)
\]  

(12)

Step 2: Based on the $j$th piece $d_j(t), P_i$ ($i=2, 3, \ldots, n+1$) obtains $D_j$ as follows:

\[
D_j = d_j(t) + R_{i,0} + R_{i,j-2}
\]  

(13)

Step 3: $P_i$ ($i=2, 3, \ldots, n+1$) keeps $D_j$ for itself, and for $j \neq i$, $P_i$ ($i=2, 3, \ldots, n+1$) encrypts $D_j$ with $K_i$ and sends $\text{Encrypt}(D_j, K_i)$ to $P_j$ ($j=2, 3, \ldots, n+1$).

Step 4: After receiving the following encrypted data from its neighboring nodes $\{P_2, P_3, \ldots, P_{i-1}, P_{i+1}, \ldots, P_{n+1}\}$ separately:

\[
\text{Encrypt}(D_{2,i}, K_{2,i})
\]

\[
\text{Encrypt}(D_{3,i}, K_{3,i})
\]

\[
\text{Encrypt}(D_{4,i}, K_{4,i})
\]

\[
\text{Encrypt}(D_{n-1,i}, K_{n-1,i})
\]

\[
\text{Encrypt}(D_{n,i}, K_{n,i})
\]

$P_i$ ($i=2, 3, \ldots, n+1$) will decrypt them and obtain $D_{2,i}, D_{3,i}, D_{4,i}, \ldots, D_{n,i}$, and then, $P_i$ ($i=2, 3, \ldots, n+1$) can obtain $\{D_{2,i}, D_{3,i}, D_{4,i}, \ldots, D_{n,i}\}$ based on the $D_i$ that it keeps for itself previously. Hence, $P_i$ ($i=2, 3, \ldots, n+1$) will obtain $\sum_{j=2}^{n+1} D_{j,i}$ and send $\text{Encrypt}(\sum_{j=2}^{n+1} D_{j,i}, K_{n,i})$ to the cluster head $P_1$.

Step 5: After receiving the following encrypted data from all its cluster members $\{P_2, P_3, \ldots, P_{n+1}\}$ separately:

\[
\text{Encrypt}(\sum_{j=2}^{n+1} D_{j,2}, K_{2,1})
\]

\[
\text{Encrypt}(\sum_{j=2}^{n+1} D_{j,3}, K_{3,1})
\]

\[
\text{Encrypt}(\sum_{j=2}^{n+1} D_{j,n+1}, K_{n+1,1})
\]

$P_1$ will decrypt them and obtain the following data $\overline{D}$ as follows:

\[
\overline{D} = \sum_{j=2}^{n+1} \sum_{i=2}^{n+1} D_{j,i} = \sum_{j=2}^{n+1} \sum_{i=2}^{n+1} D_{i,j}
\]

(14)

\[
= \sum_{i=2}^{n+1} \sum_{j=2}^{n+1} (d_{i,j} + R_{i,0} + R_{i,j-2})
\]

\[
= \sum_{i=2}^{n+1} \sum_{j=2}^{n+1} d_{i,j} + \sum_{i=2}^{n+1} \sum_{j=2}^{n+1} (R_{i,0} + R_{i,j-2})
\]

(15)

Obviously, since $\{R_{i,0}, R_{i,1}, \ldots, R_{i,n-2}\}$ are random numbers generated by $P_i$ itself, and $\{R_{2,0}, R_{3,0}, \ldots, R_{n+1,0}\}$ are random numbers sent to $P_i$ by its cluster members $\{P_2, P_3, \ldots, P_{n+1}\}$ to generate the pair-wise keys, then, the cluster head $P_1$ can easily obtain the desired data $D(t)$ without knowing the detail of $d(t)$ ($i=2, 3, \ldots, n+1$) as follows:

\[
D(t) = \overline{D} - \sum_{i=2}^{n+1} \sum_{j=2}^{n+1} (R_{i,0} + R_{i,j-2})
\]

(16)

\[
= \overline{D} - (n-1) \sum_{i=2}^{n+1} (R_{i,0} + R_{i,i-2})
\]

4. Evaluation and Analysis

In this section, we will evaluate the Cluster Based Private-Preserving Data Aggregation scheme (CP3DA) proposed in this paper. We evaluate the performance of our scheme in terms of three indicators: communication/ storage/ computational overhead, capacity of privacy-preservation, and data aggregation accuracy. Obviously, the above three indicators shall satisfy the following criteria:

- Communication, storage and computational overheads: In order to provide extra privacy-preservation, additional communication, storage, and computational overhead will be introduced. However, a good privacy-preserving data aggregation scheme shall keep these additional overheads as small as possible.

- Capacity of privacy-preservation: A good privacy-preserving data aggregation scheme shall provide good data privacy preservation for each sensor node and guarantee that each node’s data shall only be known to itself. In addition, it shall be robust to handle to some extent attacks and collusion among compromised nodes or eavesdroppers also.

- Data aggregation accuracy: A good privacy-preserving data aggregation scheme shall provide accurate aggregation of sensor data while providing good data privacy preservation for each sensor node.

4.1. Communication, Storage and Computational Overheads

The newly proposed CP3DA scheme utilizes slice based data-hiding technique and encrypted communication to protect data privacy, which will introduce some overheads in terms of communication, storage and computational as a tradeoff.

a) Communication Overhead.

For each sensor node $P_i$ in a cluster ($\alpha, \beta$) in the WSN with total $N$ different sensor nodes, suppose...
that $P$ has $N_P$ different neighboring nodes, that is, the degree of node $P$ is $N_P$, and in these $N_P$ different neighboring nodes, there are $N_P'$ different nodes with the cluster number $(\alpha, \beta)$, and the node ID of $P$ is coded as a $M$ dimensional vector $(i_1, i_2, \ldots, i_M)$, where $i_j \in \{0,1\}$ and $K=1, 2, \ldots, M$.

From the third assumption, it is easy to know that the length of the node ID of $P$ will be $M$ bits, where

$$M = \left\lceil \log_2 N \right\rceil$$ (14)

Based on these above assumptions, the communication overhead of $P$ can be estimated as follows:

Step 1: $P$ needs one round of communication to forward the Declaration Message to find out the coordinates of the Sink node, as well as the node IDs of its all neighboring nodes, and join in a cluster. Since the Declaration Message includes two fields: the first field is the coordinates of the Sink node, and the second field is the node ID of $P$. Hence, in this step, the communication overhead can be estimated as $O(L_1 + M)$, where $L_1$ is the length of the first field.

Step 2: $P$ needs one round of communication to broadcast the Hello Message to select the cluster head. Since the Declaration Message includes three fields: the first field is Node ID, the second field is Residual Energy, and the third field is Cluster Number. Hence, in this step, the communication overhead can be estimated as $O(M + L_2 + L_3)$, where $L_2$ is the length of the second field, and $L_3$ is the length of the third field.

Step 3: $P$ needs $N_P$ rounds of communication to exchange random numbers with all its $N_P$ different neighboring nodes to form pair-wise keys. Hence, in this step, the communication overhead can be estimated as $O(L_1 + N_P)$, where $L_4$ represents the maximum bits occupied by the field of random number.

Step 4: $P$ needs $N_P'$ rounds of communication to send data pieces to all its neighboring nodes having the same cluster number with itself. Hence, in this step, the communication overhead can be estimated as $O(L_1 + N_P')$, where $L_5$ represents the bits occupied by the field of data piece.

Step 5: $P$ needs one round of communication to forward its encrypted data to the cluster head. Hence, in this step, the communication overhead can be estimated as $O(L_5)$, where $L_6$ represents the bits occupied by the field of data.

From the above analysis, it is obvious that the communication overhead of the CP-DA scheme can be estimated as $O\left(\sum_{i=1}^{3} L_i + L_6 + M + \sum_{i=4}^{5} L_i + N_{max}\right)$, where $N_{max}$ represents the maximum degree of the sensor nodes in the WSN.

b) Storage Overhead.

The storage overhead of $P$ can be estimated as follows:

Step 1: $P$ needs to store the coordinates of the Sink node and itself, and the node IDs of all its neighboring nodes to know the coordinates of the Sink node and find the node IDs of its all neighboring nodes. Hence, in this step, the storage overhead can be estimated as $O(2^{*}L_1 + M*N_P)$.

Step 2: $P$ needs to store the Node ID, Residual Energy and Cluster Number of all its neighboring nodes to select the cluster head. Hence, in this step, the storage overhead can be estimated as $O(L_2 + L_3 + M*N_P)$.

Step 3: $P$ needs to store the random numbers sent by all its neighboring nodes and the random numbers generated by itself to form pair-wise keys. Hence, in this step, the storage overhead can be estimated as $O(2^{*}L_4*N_P)$.

Step 4: $P$ needs to store the pair-wise keys among itself and all its neighboring nodes. Hence, in this step, the storage overhead can be estimated as $O(L_5*N_P)$, where $L_7$ represents the maximum bits occupied by the field of pair-wise key.

Step 5: $P$ needs to store the data produced by itself and the data pieces sent by all its neighboring nodes in with the cluster number $(\alpha, \beta)$. Hence, in this step, the storage overhead can be estimated as $O(L_6 + L_7 + N_{max} + L_1 + L_6 + L_5 + N_P')$.

c) Computational Overhead.

The computational overhead of $P$ can be estimated as follows:

Step 1: $P$ needs to compute its cluster number according to the coordinates of the Sink node and itself. Hence, in this step, the storage overhead can be estimated as $O(2^{*}T_1)$, where $T_1$ denotes the computational overhead needed to execute the arithmetic operation for one time.

Step 2: $P$ needs to compare the information of its own with the information of all its neighboring nodes in order to select the cluster head. Hence, in this step, if we adopt the Quicksort algorithm, the storage overhead can be estimated as $O(2^{*}T_1*N_P + \log_2(N_P))$.

Step 3: $P$ needs to execute the hashing operation to form pair-wise keys. Hence, in this step, the storage overhead can be estimated as $O(2^{*}T_2*N_P)$, where $T_2$ represents the computational overhead needed to execute the hashing operation for one time.

Step 4: $P$ needs to slice its private data into different pieces. Hence, in this step, the storage overhead can be estimated as $O(T_3*N_P')$.

Step 5: $P$ needs to encrypt all of its data pieces with different pair-wise keys. Hence, in this step, the storage overhead can be estimated as $O(T_4*N_P')$, where $T_3$ represents the maximum computational overhead needed to execute the encrypting operation for one time.

Step 6: $P$ needs to decrypt all of the data pieces sent by its neighboring nodes. Hence, in this step, the storage overhead can be estimated as $O(T_5*N_P')$. 

153
Step 7: $P$ needs to assemble all these decrypted data pieces into an encrypted data and then send the encrypted data to the cluster head. Hence, in this step, the storage overhead can be estimated as $O(T_1*N'P+T_3)$.

Based on the above analysis, the computational overhead of the CPDA scheme can be estimated as $O(2*T_1+T_2+T_3)+N_{max}+2*T_1*N'P^*\log_2(N_{max})$.

4.2. Capacity of Privacy-Preservation

In order to evaluate the capacity of privacy-preservation in our scheme, we first define the privacy metric. In our scheme, for each sensor node $P$ in a cluster $(\alpha, \beta)$, that the private data of $P$ may be disclosed can only happen when $P$ exchange messages within the cluster and all the links between itself and its neighboring nodes within the cluster $(\alpha, \beta)$ including the cluster head are compromised. Let $p =$ probability that link is compromised, $L_{max} =$ the maximum cluster size, $L_{min} =$ the minimum cluster size, $k =$ cluster size, $P(k=m) =$ probability that a cluster size is $m$, then the capacity of privacy-preservation in our scheme can be estimated as,

$$p = \sum_{m=L_{min}}^{L_{max}} P(k = m) (1 - (1 - p^m)^m) (15)$$

Especially, when a cluster has two sensor nodes (with one cluster head and one cluster member) only, then, the cluster head will know the private data of the cluster member. To avoid the above scenario, then, we shall do our best to guarantee that there is $L_{min} > 2$.

Considering that the total $N$ sensor nodes are deployed evenly in the WSN with the length of $L$ and width of $W$, then, it is easy to know that the WSN can be divided into $\lceil W * L / (R^2 / 2) \rceil$ different clusters. And then, In order to guarantee that there is $L_{min} > 2$, it is obvious that we shall guarantee that the following Formula (16) to be true,

$$N/\lceil W * L / (R^2 / 2) \rceil > 2$$

4.3. Data Aggregation Accuracy

From the description given in previous sections, as we can see, in ideal situations, when there is no data loss in the WSN, it is obvious that the Sink node will obtain 100 % accurate aggregation results, since there is no data loss during the process of data aggregation within each cluster when adopting our CPDA scheme.

However, due to collisions over wireless channels and processing delays, messages may get lost or delayed in a WSN, which will impact the data aggregation accuracy. Due to the fact that larger epoch duration will provide data packets with a better chance of being delivered before the timeout, which means, larger epoch duration leads to better accuracy. Therefore, in order to keep good data aggregation accuracy, we shall select suitable epoch duration when adopting our newly given CPDA scheme.

4.4. Simulation Results

To further evaluate the performance of the newly proposed CPDA scheme, in this section, we provide extensive simulations to compare our scheme with CPDA proposed by He et al. [8]. For convenience, in the experiments, we adopt the following scenario similar to CPDA: There are 600 sensor nodes deployed evenly over a 400x400 square area. And for each sensor node, its valid communication radius is 50, its data rate is 1 Mbps, and its original energy is 100.

The size of the random numbers generated by each node is restricted in the range of 0–1024. And for each sensor node $P$ in a cluster $(\alpha, \beta)$, it will slice its data equally into $N'P$ pieces, where $N'P$ is the number of neighboring nodes of $P$ that have the same cluster number $(\alpha, \beta)$ with $P$.

From the previous introduction, we can easily know that there are $N=600$, $R=50$, $M=10$. And when adopting the same coding technique with node ID, then, we will have $L_1=9+9=18$, $L_2=7$, $L_3=10$, $N_{max} \in [20,60]$, $L_1*N'P \approx 1 M$, and $L_0 \approx 1 M$. Therefore, it is easy to know that the communication overheads of the CPDA scheme will equal to 2M approximately.

Fig. 2 illustrates the comparison of the communication overhead of CPDA and CPDA with $p_r = 0.3$ under different epoch durations. In the simulation, we use the total number of bytes of all packets transmitted during the aggregation in a randomly selected cluster as the metric, and each point in the Fig. 2 is the average result of 50 runs of the simulation.

![Fig. 2. Comparison of the communication overheads of CPDA and CPDA with $p_r = 0.3$.](image-url)
be a little higher than that of the CPDA scheme. The main reason is that the private data in each sensor node is required to be sliced into data pieces and these data pieces are required to be sent to all sensor nodes belonging to the same cluster in our scheme.

Fig. 3 illustrates the comparison of the storage overhead of CP\(^2\)DA and CPDA with \(p_c = 0.3\).

According to Fig. 3, as we can see, the storage overheads of our scheme CP\(^2\)DA will be much smaller than that of the CPDA scheme. The main reason is that each sensor node needs only to store the data produced by itself and the data pieces sent by all its neighboring nodes having the same cluster number with it by using our CP\(^2\)DA scheme, while, in CPDA, each sensor node needs to store the data produced by itself and all its neighboring node with the same cluster number.

Fig. 4 illustrates the comparison of the computational overhead of CP\(^2\)DA and CPDA with \(p_c = 0.3\). For convenience of comparison, we consider the time costs of data aggregation within the cluster only in the simulation.

According to Fig. 4, as we can see, the computational overheads of our scheme CP\(^2\)DA will be much smaller than that of the CPDA scheme. This is due to that the matrix operation is not required in our scheme, while it is absolutely necessary in CPDA scheme.

Fig. 5 illustrates the comparison of the capacity of privacy-preservation of CP\(^2\)DA and CPDA with \(p_c = 0.3\).

By observing Fig. 5, the probability of privacy compromised in CPDA has much steeper slope than that of our CP\(^2\)DA scheme. It is due to that a new clustering scheme and slice based privacy-preserving scheme are adopted in our scheme, while, in CPDA scheme, the private data are not divided into data pieces to protect its privacy.

Fig. 6 illustrates the variation of data aggregation accuracy under different epoch durations when adopting our newly given CP\(^2\)DA scheme.

By observing Fig. 6, the data aggregation accuracy of our CP\(^2\)DA scheme increases as the epoch duration increases. The main reason is that with larger epoch duration, the data packets to be sent within this duration will have less chance to collide due to the extended average packet sending intervals.
5. Conclusions

In resource-limited and power-constrained wireless sensor networks, it is a challenging work to design efficient privacy-preserving data aggregation protocols. In this paper, by adopting a new slice-based technique, we present a cluster-based privacy-preserving data aggregation protocol, and compare the performance with a typical data aggregation scheme CPDA. Simulation results and theoretical analysis show the efficacy of our newly given scheme.

Acknowledgements

The authors thank the anonymous referees for suggestions that helped improve the paper substantially. And the project is partly sponsored by the Colleges and Universities Open Innovation Platform Fund of Hunan Province (No. 13K041), the Hunan Provincial Natural Science Foundation of China (No. 14JJ2070), the construct program of the key discipline in Hunan province, the State Education Ministry Scientific Research Foundation for the Returned Overseas Chinese Scholars, and the Introduced Talent Start-up Fund Project of Xiangtan University.

References


