Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2013, Article ID 241261, 14 pages
http://dx.doi.org/10.1155/2013/241261

Research Article

ID List Forwarding Free Confidentiality Preserving Data Aggregation for Wireless Sensor Networks

Liehuang Zhu, Zhen Yang, Mingzhong Wang, and Meng Li

Beijing Engineering Research Center of Massive Language Information Processing and Cloud Computing Application, School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China

Correspondence should be addressed to Liehuang Zhu; liehuangz@bit.edu.cn

Received 10 January 2013; Accepted 26 March 2013

Academic Editor: Xu Yongjun

Wireless sensor networks (WSNs) are composed of sensor nodes with limited energy which is difficult to replenish. Data aggregation is considered to help reduce communication overhead with in-network processing, thus minimizing energy consumption and maximizing network lifetime. Meanwhile, it comes with challenges for data confidentiality protection. Many existing confidentiality preserving aggregation protocols have to transfer list of sensors' ID for base station to explicitly tell which sensor nodes have actually provided measurement. However, forwarding a large number of node IDs brings overwhelming extra communication overhead. In this paper, we propose provably secure aggregation scheme perturbation-based efficient confidentiality preserving protocol (PEC2P) that allows efficient aggregation of perturbed data without transferring any ID information. In general, environmental data is confined to a certain range; hence, we utilize this feature and design an algorithm to help powerful base station retrieve the ID of reporting nodes. We analyze the accuracy of PEC2P and conclude that base station can retrieve the sum of environmental data with an overwhelming probability. We also prove that PEC2P is CPA secure by security reduction. Experiment results demonstrate that PEC2P significantly reduces node congestion (especially for the root node) during aggregation process in comparison with the existing protocols.

1. Introduction

Wireless sensor networks (WSNs) integrate microelectromechanical systems (MEMSs) technology, sensor technology, and communication technology. WSN can sense, transport and process different environmental data in its deployment area by hundreds of sensor nodes with limited computation and energy capacities. WSNs have been extensively used in military surveillance, environmental monitoring, production control, and real-time traffic monitoring [1].

Because WSNs are usually deployed in remote, unattended, or even hostile environment, the energy of sensor nodes is not easy to get replenished. Hence, how to reduce energy cost and prolong the network lifetime has become key issues for WSNs [2, 3]. It is generally believed that power consumption of each sensor node tends to be dominated by data transmission. According to [4], energy cost of transmitting a single bit of data is equivalent to that of 800 instructions. Data aggregation [2, 5] mechanisms avoid transmitting environmental data through in-network process of summarizing and combining sensor data, thus reducing the amount of data transmission and effectively maximizing network lifetime.

Data confidentiality [6–11] is crucial in many WSN applications, like military surveillance. If data confidentiality is compromised, the sensitive information collected will be leaked to adversary. However, there is a conflict between data aggregation and data confidentiality protocols [12]: data aggregation prefers to operate on plain data and confidentiality protection requires data to be encrypted. Extensive secure data aggregation research [6–11, 13–15] has been conducted. Data aggregation protocols usually cannot operate on encrypted data such that intermediate node has to decrypt packets received from downstream, aggregate the plaintext data with its own, encrypt the aggregated result, and forward to upstream.

Two common approaches to preserve data confidentiality without decryption/encryption are homomorphic

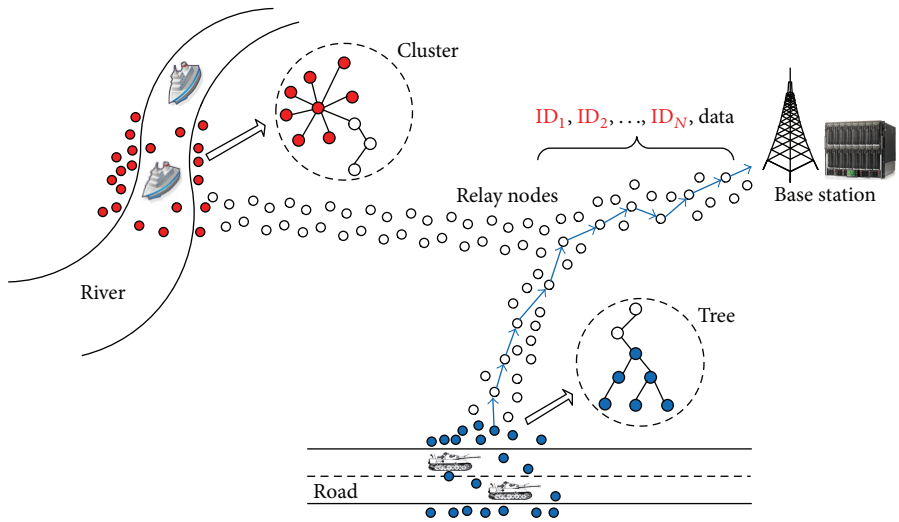


Figure 1: An example of environmental surveillance system in battlefield.

encryption [7, 8, 10], and secret perturbation [6, 9, 11]. Homomorphic encryption is an encryption transformation that allows direct computation on encrypted data. However, end-to-end security of symmetric homomorphism [7] is easily compromised if any node is corrupted and the computational cost and communication overhead of asymmetric homomorphism [8, 10] are not preferable. In comparison, secret perturbation-based schemes add a perturbation to the value of each reporting node using shared secret key with base station (*BS*). *BS* retrieves the final aggregation result by removing all these perturbation. Since the key shared between each node and *BS* is unique, adversary will not compute other nodes' sensed data or intermediate aggregation result if one key is compromised.

BS has to know which sensor nodes have provided measurement before it can correctly remove the perturbations brought by these sensor nodes. A straightforward solution is to require every node participating/not participating in aggregation process to report its ID, according to the proportion of nodes satisfying *BS*'s query.

However, this approach may bring high extra overhead. Feng et al. [9] proposed a family of secret perturbation-based schemes that can protect sensor data confidentiality while trying to minimize the number of ID to be transferred. In FSP scheme, every sensor node must reply a perturbed actual or dummy data item, no matter the node has satisfying data or not. *BS* will simply subtract hash value for every sensor node to compute final aggregation result, and communication overhead caused by ID transmission is avoided. However, it requires all sensor nodes to report data no matter whether they have data satisfying the query. This may result in high extra communication overhead when only a small number of sensor nodes have data to report and communication overhead caused by extra perturbed data can be much larger than that of forwarding ID. Hence, in their ideal scheme O-ASP, aggregating node first has to compute whether overhead of transmitting ID and perturbed data or overhead of transmitting all perturbed data is larger. Either way, O-ASP endures high communication overhead, and it is unrealistic for each sensor node to know the membership and topology of the whole network, and it knows whether each of these nodes has data satisfying each particular query.

Moreover, the transmission of nodes' ID makes [9, 11] not suitable for the scenario shown in Figure 1, where we want to monitor the activities of tanks and battleships, and there is a long path to travel through before aggregation result gets to *BS*. To achieve this goal, a cluster or a tree of sensor nodes is deployed in the battlefield, while *BS* is in a secure location away to collect data reported by sensors. All data has to be forwarded on a long path from the targeted area to *BS*. For [9, 11], ID list is transmitted such that the energy is wasted on the long path, and "single point of failure" could happen if there are not enough nodes on such path. The application scenarios in military surveillance also include the case that US army uses REMBASS to collect data (like ground motion, sound, infrareds, and magnetic fields) and forward the aggregation result to command center. PEC2P fits in this scenario and does not have any requirement on the type of data.

In this paper, we present perturbation-based efficient confidentiality preserving protocol (PEC2P) which can protect data confidentiality without transmitting any ID information. Generally, we use one-way hash function as perturbation added to the environmental data. Since *BS* usually has powerful computational capability in WSNs, we propose to trade computation consumption at *BS* for energy cost of sensors and introduce a new approach for *BS* to compute and tell which nodes have actually sensed data and contributed to the aggregation process after receiving the final aggregation result. Our approach specifically fits for scenarios where aggregation result has to travel a long path before arriving at *BS*. In summary, contribution of this paper includes the following.

(1) We draw attention to the ID-list transmitting problem in WSNs and propose the first approach which does

not require forwarding any node ID but computing and selecting by *BS*. As a result, communication overhead is reduced and reporting nodes' information is further hidden.

(2) We avoid using the random number r verified by commonly applied authenticated broadcasting, thus reducing network delay. Instead, we update the secret key of all reporting nodes after each data aggregation to keep indistinguishability from adversary.

(3) We prove that our protocol is CPA secure by security reduction.

(4) We measure the performance of our protocol through both theoretic analysis and experiments on TinyOS [16]. We analyze the accuracy of PEC2P and compare its communication overhead with existing protocols.

2. Related Work

Girao et al. proposed CDA [7] using symmetric key-based privacy homomorphic encryption. In their approach, sensor nodes share a common symmetric key with the *BS* which is hidden from aggregators, and aggregators can perform aggregation functions directly on the ciphertext instead of carrying out costly decryption and encryption operations. Symmetric homomorphism has the advantage of fast computation. However, secret key is shared among all nodes such that data confidentiality is lost once a sensor node and its shared key are compromised.

Mykletun et al. [8] investigated several additive homomorphic public-key encryption schemes and their applicability to WSNs. In general, these schemes preload public key in sensor nodes and aggregate encrypted data. Then *BS* can decrypt aggregation result by its secret key. Albath and Madria [10] proposed an ECC-ElGamal based homomorphic encryption scheme to achieve confidentiality for in-network aggregation in wireless sensor networks. Even if the adversary compromises a node and obtains the public key, it cannot obtain the plaintext of intermediate aggregation results. Hence, public key-based homomorphic encryption schemes are resilient to node compromise attacks. However, the computational cost and communication overhead of public key encryption scheme are not quite tolerable for WSNs, especially when sensors are collecting diverse statistics (like temperature, humidity, and pressure).

Castelluccia et al. [6] first proposed an additively homomorphic encryption scheme which simply adds secret key k to environmental data x as ciphertext $c = x + k$. Each node has a unique secret key such that one node's corruption does not affect the data confidentiality of other nodes. Castelluccia et al. [11] improved their scheme in [11] by proposing a simple and provably secure encryption scheme that allows efficient additive aggregation of encrypted data. Each reporting node i encrypts plaintext data m_i as: $c_i = m_i + h(f_{ek_i}(r))$. The security of their scheme is based on the indistinguishability property of a pseudorandom function (PRF). However, ID-list of sensors has to be transferred and cannot be aggregated.

Feng et al. [9] tried to alleviate the ID-list problem and proposed a family of secret perturbation-based schemes that can protect sensor data confidentiality without disrupting the additive data aggregation result. BSP and FSP are two basic schemes which take nonredundant reporting approach and fully reporting approach, respectively. The ideal scheme O-ASP assumes that each sensor node knows the membership and topology of the whole network, and it knows whether each of these nodes has data satisfying each query. Then, *BS* computes aggregation and communication cost of two approaches for each cell before selecting one. To overcome the unrealistic assumption, D-ASP is proposed to enable nodes to make decisions based only on their locally available information, and interactions only take place within a cell or between neighboring cells. However, it is difficult for nodes to decide whether to report their ID with locally available information and it makes no difference when the number of reporting nodes is the same as nonreporting nodes. It also causes extra communication cost and network delay for waiting and deciding.

PRDA [15] pointed out that the transmission of sensor node IDs along with aggregated data packets increases the communication overhead of the network. Therefore, it keeps a table that consists of sensor node IDs and their corresponding small index numbers in each data aggregator. After the cluster forming, data aggregator generates the index table and sends it to *BS*. During data aggregation, instead of sending 2-byte sensor node IDs, data aggregators send corresponding index numbers. *BS* can find the ID of sensor nodes in the index table. However, index numbers are only used within clusters.

Although existing schemes tried to reduce the amount of IDs, they still suffer from related communication cost, and dropping ID or sending false ID will lead *BS* to compute false aggregation result.

Our work requires no ID to be forwarded and achieves a good trade-off between confidentiality and efficiency by adopting perturbation. With this improvement, we manage to simultaneously preserve data confidentiality and significantly reduce overall communication overhead, avoiding high energy consumption in aggregation phase.

3. System Model

3.1. Network Assumption. We assume a multilevel sensor network tree that consists of N (less than 1000) sensor nodes and certain amount of relay nodes. Sensor nodes are deployed in areas of interest, and they can sense and aggregate data. Both tree and cluster topologies can be applied in aggregation structure. In this paper, we use aggregation tree to illustrate our protocol. Aggregation tree could be formed as in TAG [4]. Relay nodes just forward messages, and they consist of a long path from targeted areas to *BS*. The powerful *BS* with transmission range covering the whole network is capable of broadcasting messages to all nodes directly. Each sensor node has a unique ID picked from the set $\{0, 1, \ldots, N-1\}$. After the aggregation tree is formed, each sensor node monitors its surrounded environment to generate environmental data which is an integer ranging from $[v_{\min}, v_{\max}]$. Environmental data (e.g., temperature) can be

converted to integers if necessary. Each reporting node and aggregator sends their messages up the aggregation tree. The message has the following format:

$$\langle c, hax \rangle, \quad (1)$$

where c is the number of reporting nodes in network and hax is the sum of environmental data and perturbation.

3.2. Design Goals. When designing confidentiality protection schemes, we aim to achieve the following goals.

(1) Data accuracy: *BS* can correctly retrieve the sum of environmental data with an overwhelming probability.

(2) Data confidentiality: the aggregation result should only be known by *BS* and PEC2P is CPA secure.

(3) Efficiency: the protocol should help to reduce communication overhead and prolong the network lifetime.

Definition 1 (Chosen Plaintext Attack). In this attack, the adversary has the ability to obtain the encryption of plaintexts of its choice. It then attempts to determine the plaintext that was encrypted in some other plaintext [17].

Definition 2 (Negligible Function). A function F is negligible if for every polynomial $p(\cdot)$, there exists an N such that for all integers $n > N$, it holds that $F(n) < 1/p(n)$. An equivalent formulation of the above is to require that for all constants c, there exists an N such that for all $n > N$, it holds that $F(n) < n^{-c}$.

We define an experiment for any private-key encryption scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$, any adversary A, and any value n of security parameter.

The CPA Indistinguishability Experiment $PriK^{\text{CPA}}_{A,\Pi}(n)$.

(1) A random key k is generated by running $\text{Gen}(n)$.

(2) The adversary A is given input 1^n and oracle access to $\text{Enc}_k(\cdot)$, and outputs a pair of messages m_0, m_1 of the same length.

(3) A random bit $b \leftarrow \{0, 1\}$ is chosen, and then a ciphertext $c \leftarrow \text{Enc}_k(m_b)$ is computed and given to A. We call c the challenge ciphertext.

(4) A continues to have oracle access to $Enc_k(\cdot)$, and outputs a bit b'.

(5) The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise. (A succeeds if $PriK^{\text{CPA}}_{A,\Pi}(n) = 1$).

Definition 3 (CPA secure). A private-key encryption scheme $\Pi = (\text{Gen}; \text{Enc}; \text{Dec})$ has indistinguishable encryptions under a chosen-plaintext attack (or is CPA secure) if for all probabilistic polynomial-time adversaries A there exists a negligible function negl such that:

$$Pr\left[PriK^{\text{CPA}}_{A,\Pi}(n) = 1\right] \leq \frac{1}{2} + \text{negl}(n). \quad (2)$$

3.3. aAttacker Model. We assume the existence of a global probabilistic polynomial time (PPT) adversary, which can choose to compromise a small subset of nodes and obtain all secrets of these nodes. With oracle access, it can also obtain the ciphertext for any chosen plaintext from any of the uncompromised nodes. Once the adversary compromises a sensor node, it will obtain its secret key and may modify, forge or discard messages, or simply transmit false aggregation results.

In this paper, we do not consider stealthy attacks [18] where the attacker's goal is to make the *BS* accept false aggregation results while not being detected. Also, we do not consider the denial-of-service (DoS) attack in various protocol layers [19, 20] where the adversary prevents the querier from getting any aggregation result at all. However, if a node does not respond to queries, it is clear that something is wrong, and solutions can be implemented to remedy this situation. Sybil/node replication attacks [21] or "wormhole" formation [22, 23] are beyond the scope of this paper.

4. PEC2P

The proposed scheme PEC2P mainly consists of bootstrapping phase, data aggregation phase, and result retrieving phase.

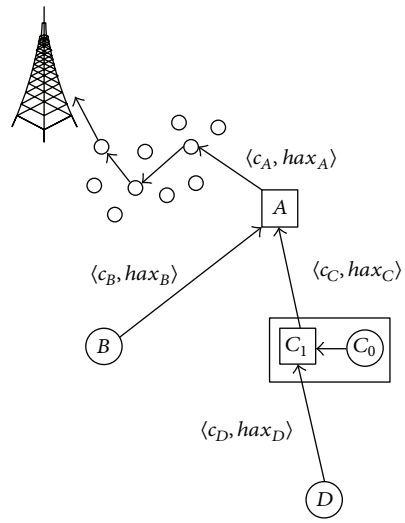
4.1. Bootstrapping Phase. In bootstrapping phase, modulus $M = 2^l$ is stored in all nodes, and so is a collision-resistant cryptographic hash function $H : \{0,1\}^* \leftarrow \{0,1\}^l$ and a PRF $f : \{0,1\}^l \leftarrow \{0,1\}^l$.

We further assume that *BS* first runs Algorithm 1 such that a unique initialization vector IV_i is generated, and secret key $k_i = IV_i$ is stored in *BS*'s local record and node i.

4.2. Data Aggregation Phase. Each sensor node in targeted area may behave as a sensing node, an aggregator, or combined. To simplify the discussion, we assume that each node can perform one role of sensing or aggregating without the loss of generality. Any node with combined role can be logically split into a sensing node and an aggregating node. As shown in Figure 2, aggregator C both senses data and aggregates data from downstream. It is divided into sensing node C_0 and aggregating node C_1. After the transformation, only leaf nodes sense environmental data.

In aggregation phase, when a targeted event happens or *BS* disseminates a query, each leaf sensor node i with environmental data x_i runs Algorithm 2 to compute individual aggregation result $\langle c_i, hax_i \rangle$. First, i inputs environmental data x_i, then sets $c_i = 1$ and $hax_i = x_i + H(k_i)$ since it has no children nodes. Second, i forwards the result to its parent node for data aggregation. Finally, i updates its secret key $k_i = f(k_i)$. Other leaf senor nodes remain hibernated.

During each aggregation, upon receiving a message from one of its children nodes for the first time, each aggregator i starts a timer t and collects other messages before t fires. Then, it runs Algorithm 3 to compute partial aggregation result $\langle c_i, hax_i \rangle$. First, i computes partial count $c_i = \sum_{j \in S_i} c_j$ and partial perturbed data $hax_i = \sum_{j \in S_i} hax_j \mod M$. Then



$x_D = 15$
$c_D = 1$
$hax_D = x_D + H_D \bmod M$
$= F9\ 37\ C3\ 8D$

$x_C = 10$
$c_C = 1 + C_D$
$hax_C = x_C + H_C + hax_D \bmod M$
$= 26\ F7\ 08\ 3D$

$x_B = 21, c_B = 1$
$hax_B = x_B + H_B \bmod M$
$= A4\ 6F\ E0\ D9$

$c_A = c_B + c_C = 3$
$hax_A = hax_B + hax_C \bmod M$
$= CB\ 66\ 19\ 16$

Figure 2: An example of data aggregation phase.

begin
randomly pick master key $SK \in \{0,1\}^l$;
for $i \leftarrow 0$ to $N-1$ **do**
$IV_i \leftarrow f_{SK}(i)$;
store IV_i in BS;
store $k_i = IV_i$ in BS and node i;
end

Algorithm 1: Bootstrapping algorithm.

begin
Input: environmental data x_i;
$c_i \leftarrow 1$;
$H_i \leftarrow H(k_i)$;
$hax_i \leftarrow (x_i + H_i) \bmod M$;
$k_i \leftarrow f(k_i)$;
return $\langle c_i, hax_i \rangle$;
end

Algorithm 2: Perturbation algorithm.

begin
$c_i \leftarrow \sum_{j \in S_i} c_j$;
$hax_i \leftarrow \sum_{j \in S_i} hax_j \bmod M$;
return $\langle c_i, hax_i \rangle$;
end

Algorithm 3: Aggregation algorithm.

i forwards the result to its parent node. Aggregators receiving no messages from downstream just remain hibernated. Note that we count number to trace the contributing nodes in BS; hence, synchronization among sensors is not needed.

Definition 4. S_i is a set of reporting node's ID, and these nodes are node i's children nodes.

To show how our scheme works, we take Figure 2 as an example. Node B and D are leaf sensor nodes with their own environmental data x_B and x_D. Node C is divided into C_0 and C_1 such that C_0 runs Algorithm 2 and node C_1 runs Algorithm 3 respectively. Aggregator A just forwards messages after aggregating data received from B and C. BS obtains the final aggregation result: $C_{BS} = 3$ and $HAX_{BS} = 0xCB66E916$.

4.3. Result Retrieving Phase. In result retrieving phase, after receiving final aggregation result $\langle C_{BS}, HAX_{BS} \rangle$, BS runs Algorithm 7 to retrieve ID list and actual aggregation result. First, BS orderly selects a list IDL of C_{BS} nodes and corresponding shared keys k_j from the N nodes, and BS computes

$$Agg = \left(HAX_{BS} - \sum_{i \in IDL} H(k_i) \right) \bmod M \tag{3}$$

if $Agg \in [C_{BS} * v_{\min}, C_{BS} * v_{\max}]$, and then BS will admit that Agg is the actual aggregation result $\sum_{i \in IDL} x_i$ and update secret keys for the found C_{BS} nodes. If not, BS will continue searching.

To improve searching efficiency for BS, we can first divide the network into clusters of trees each containing part of N nodes. Further analysis is in Section 5.3.

/* Define a private-key encryption scheme for messages of length L and key of length n as follows: */

(i) **Gen:** on input 1^n, choose $k \leftarrow \{0,1\}^n$ uniformly at random and output it as the key.

(ii) **Enc:** on input a key $k \leftarrow \{0,1\}^n$ and a message $m \leftarrow \{0,1\}^L$, output the ciphertext:

$$\langle c = 1,\ s = H(k)\rangle$$

(iii) **Dec:** on input a key $k \leftarrow \{0,1\}^n$ and a ciphertext $\langle c, s\rangle$, search for matching set S and output the plaintext:

$$m = s - \sum_{i \in S} H(k_i)$$

(iv) **Addition of Ciphertext:** given two ciphertext $\langle c_i, s_i\rangle$ and $\langle c_j, s_j\rangle$, output $\langle c_l, s_l\rangle$ as aggregation ciphertext:

$$c_l = c_i + c_j$$
$$s_l = \left(s_i + s_j\right) \bmod M$$

Algorithm 4: Construction Π^*.

5. Analysis and Experiments

5.1. Accuracy Analysis

Theorem 5. *PEC2P has a probability of at least*

$$1 - \frac{C_{BS} * \left(v_{\max} - v_{\min}\right)}{M - 1 - C_{BS} * v_{\min}} \tag{4}$$

in finding the correct combination in result the retrieving phase, given the environmental data in the range $[v_{\min}, v_{\max}]$, *the hash value in the range* $[0, 2^l - 1]$, *and modulus* M *is* 2^l.

Proof. We assume that $\{t \leftarrow \{0,1\}^\lambda : H(t)\}$ is the uniform distribution over $\{0,1\}^\lambda$, and then $H(x)$ is independent of $H(y)$ $(x \neq y)$. The probability of $H(x) = H(y)$ is $1/2^\lambda$. When $c = N$: $|T_1| = |T_2| = c$, $T_1 \neq T_2$, then we believe the probability of $\sum_{i \in T_1} \mathrm{Hash}(k_i) = \sum_{j \in T_2} \mathrm{Hash}(k_j)$ is $N/2^\lambda$.

Thus, adding C_{BS} environmental data together will result in a number in the range $[C_{BS} * v_{\min}, C_{BS} * v_{\max}]$, and the aggregation result is in the range $[C_{BS} * v_{\min}, C_{BS} * v_{\max} + M - 1] = [C_{BS} * v_{\min}, M - 1]$. If the result is valid, it has to belong to range $[C_{BS} * v_{\min}, C_{BS} * v_{\max}]$. Then, the probability that *BS* accepts a false aggregation result is at most

$$\frac{C_{BS} * \left(v_{\max} - v_{\min}\right)}{M - 1 - C_{BS} * v_{\min}}. \tag{5}$$

Hence, (4) holds. □

If we have 1024 nodes in the network and the data sensed from the environment is in the range $[0, 2^{32} - 1]$, we use SHA-1 as our hash function, and the output is in the range $[0, 2^{160} - 1]$. We can calculate the probability that *BS* accepts a false aggregation result is 2^{-118} which can be ignored.

We have implemented PEC2P using simple WSN experimental system to sense temperature in lab. Characteristics of SimpleWSN node is shown in Table 1.

Results are shown in Table 2. *BS* has ID $'01'$, and sensor node's ID $\in \{'01', '02', '03', '04', '05'\}$. Column 1 displays the number of participating nodes C from aggregation result $\langle C, HAX\rangle$. Column 2 displays the perturbed data HAX from $\langle C, HAX\rangle$. Column 3 displays the sum of hash value computed by *BS*. Column 4 displays the sum of environmental data after *BS* searching and subtracting the sum of hash value from HAX. The IDs of found nodes are shown in column 5. The temperature sensed is hexadecimal integer. We use Temperature (°C) $= ((t/4096) * 1.5 - 0.986)/(0.00355)$, provided by the SimpleWSN experimental platforms, to transform environment data to floating-point number which represent the Celsius degree. The average temperature is about 30 degrees Celsius in our experiment. The results justified the accuracy of PEC2P such that if we subtract data in column 3 from data in column 2, we will end up with data in column 4. The results verified that both the exact IDs and actual aggregation result are retrieved correctly.

Table 1: Characteristics of simple WSN node.

CPU 8-bit	8 MHz
Storage	10 Kbytes RAM 48 Kbytes FLASH
Communication	2.4 GHz
Bandwidth	250 Kbps
Operating system	TinyOS

5.2. Security Analysis.

We assume that each sensor node shares a unique key with *BS* and a common one-way hash function H is used. When an event happens, all nodes which are collecting environmental data will add the hash value computed on $f(k)$ to the environmental data x. Intuitively, since key k_i is only shared between node i and *BS*, other node j $(j \neq i)$ cannot successfully compute $H(k_i)$ with the probability ϵ that is not negligible. And it is also difficult for adversaries to compute the correct hash value of any given x. Hence, both privacy and confidentiality are achieved. We will prove this by security reduction. First, we construct an encryption scheme (Algorithm 4).

Table 2: Results of *BS* running selection algorithm after receiving aggregation results.

C	*HAX*				Hash sum				Raw Data				ID list				
03	50	63	46	27	50	63	23	3D	00	00	22	EA	00	03	00	05	06
03	49	77	C2	A4	49	77	9F	8F	00	00	23	15	00	03	04	00	06
02	56	A4	A8	A3	56	A4	91	38	00	00	17	6B	02	00	04	00	00
04	E1	1D	4A	91	E1	1D	1B	CA	00	00	2E	C7	02	03	04	00	06
01	2D	48	11	FF	2D	48	06	4B	00	00	0B	B4	00	00	04	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
04	BD	98	F6	19	BD	98	C7	8A	00	00	2E	8F	02	00	04	05	06
02	ED	48	FF	EE	ED	48	E8	78	00	00	17	76	00	03	04	00	00
03	A3	A6	20	EB	A3	A5	FD	F7	00	00	22	F4	02	00	04	05	00
01	5B	AD	89	B0	5B	AD	7D	EB	00	00	0B	C5	00	03	00	00	00
03	AE	AF	C4	01	AE	AF	A0	FF	00	00	23	02	02	03	00	05	00
01	EC	09	B6	05	EC	09	AA	6B	00	00	0B	9A	00	00	00	00	06
02	DB	23	9A	C9	DB	23	83	76	00	00	17	53	02	00	00	00	06
03	15	8B	95	E4	15	8B	72	ED	00	00	22	F7	02	00	04	05	00
02	16	E3	FB	2A	16	E3	E3	EC	00	00	17	3E	00	00	04	05	00
01	64	91	C9	E1	64	91	BE	29	00	00	0B	B8	02	00	00	00	00

Lemma 6. *Algorithm 4 is CPA secure if H is a pseudorandom function (PRF). One has*

$$Pr\left[PriK_{A,\Pi^*}^{CPA}(n) = 1\right] \leq \frac{1}{2} + negl(n). \tag{6}$$

Proof. If we replace the hash function H in Algorithm 4 with a truely random function F, we can have a new construction Π'. It is obvious that

$$Pr\left[PriK_{A,\Pi'}^{\mathrm{CPA}}(n) = 1\right] \leq \frac{1}{2} + \mathrm{negl}(n). \tag{7}$$

If H fulfills the requirement, then $\{t \leftarrow \{0,1\}^{\lambda} : H(t)\}$ is the uniform distribution over $\{0,1\}^{\lambda}$. Therefore, (6) holds. □

Theorem 7. *PEC2P is secure against CPA hash function if the following distributions are to be identical:*

$$\left\{t \leftarrow \{0,1\}^{\lambda} : H(t) + m_0\right\}, \left\{t \leftarrow \{0,1\}^{\lambda} : H(t) + m_1\right\}. \tag{8}$$

Proof. Proof for the nonhashed scheme. we assume that adversary A attacks (CPA) PEC2P with success probability $(1/2) + \epsilon(n)$. Now, we can construct a fast algorithm A' to "break" Construction Π^*, and A' tries to achieve its goal by running A as in Algorithm 5.

$$\begin{aligned}
&Pr_{A'}^{H}[\text{Success}] \\
&= \frac{1}{2}\left\{Pr\left[b'' = 0 \mid b = 0\right] + Pr\left[b'' = 1 \mid b = 1\right]\right\} \\
&= \frac{1}{2}\left\{\frac{1}{N}Pr\left[b'' = 0 \mid b = 0, b' = 0\right]\right. \\
&\qquad + \frac{N-1}{N}Pr\left[b'' = 0 \mid b = 0, b' = 1\right] \\
&\qquad + \frac{1}{N}Pr\left[b'' = 1 \mid b = 1, b' = 1\right] \\
&\qquad \left. + \frac{N-1}{N}Pr\left[b'' = 1 \mid b = 1, b' = 0\right]\right\} \\
&= \frac{1}{2}\left\{\frac{1}{N}Pr\left[PriK_{A,\mathrm{PEC2P}}^{\mathrm{CPA}}(n) = 1\right] + \frac{N-1}{N} * \frac{1}{2}\right. \\
&\qquad \left. + \frac{1}{N}Pr\left[PriK_{A,\mathrm{PEC2P}}^{\mathrm{CPA}}(n) = 1\right] + \frac{N-1}{N} * \frac{1}{2}\right\} \\
&= \frac{N-1}{N} * \frac{1}{2} + \frac{1}{N}Pr\left[PriK_{A,\mathrm{PEC2P}}^{\mathrm{CPA}}(n) = 1\right] \\
&= \frac{1}{2} + \frac{1}{N}\left(\frac{1}{2} + \epsilon(n)\right) = \frac{1}{2} + \frac{\epsilon(n)}{N}.
\end{aligned} \tag{9}$$

/* A' tries to break $Enc_k(x) = x + H(k)$ */
(1) A' initiates other $N - 1$ nodes and has access to N oracle $Enc(\cdot)$.
(2) A implements *PEC2P* l times and obtain the ciphertext of message x_i $(i = 1, 2, \ldots, l)$.
(3) A' forwards the queries to the network and return $H(f(k_i))$ to A.
(4) A outputs two messages m_0, m_1, sending them to A'.
(5) A random bit $b \leftarrow \{0, 1\}$ is chosen and A' makes an encryption query for m_b to $Enc_k(\cdot)$ and get back challenge ciphertext c_b $(b \in \{0, 1\})$.
(6) If c_b is from the node which holds secret key k, then A' returns c_b to A.
(7) A output a bit b' and returns it to A'.
(8) A' outputs $b'' = b'$.
(9) Else A' outputs $b'' = 0$ with the probability of 1/2 and outputs $b'' = 1$ with the probability of 1/2.
(10) Output 1 if $b'' = b$ and output 0 otherwise.

Algorithm 5: A'.

Input: $\langle IDList, HAX_{BS}\rangle$:
begin
  $Agg \leftarrow -1$;
  while $IDL \neq \perp$ **do**
    $hax_{BS} \leftarrow 0$; $j \leftarrow 0$;
    for $i \leftarrow 0$ to $N - 1$; **do**
      if $IDL[i] = 1$ **then**
        $hax_{BS} \leftarrow hax_{BS} + H(k_{ri})$;
    if $HAX_{BS} - hax_{BS} \in [C_{BS} * v_{\min}, C_{BS} * v_{\max}]$
    then
      $Agg \leftarrow HAX_{BS} - hax_{BS}$;
    for $j \leftarrow 0$ to $C_{BS} - 1$ **do**
      $k_{\text{temp}[j]} \leftarrow f(k_{\text{temp}[j]})$;
    break;
    return Agg;
  return Agg;
end

Algorithm 6: Matching algorithm.

According to Lemma 6, we should have

$$\frac{1}{2} + \frac{\epsilon(n)}{N} \leq \frac{1}{2} + \text{negl}(n). \tag{10}$$

Therefore, $\epsilon(n) \leq \text{negl}(n)$.

Security of the Hashed Version. Only a few modifications to this security proof are needed in order to prove the security of the hashed variant. First, in Algorithm 5, all ciphertext are of now generated using the hashed values of k. Second, the security proof of the hashed scheme relies on the fact that $\{t \leftarrow \{0, 1\}^\lambda : H(t) + m_0\}$ and $\{t \leftarrow \{0, 1\}^\lambda : H(t) + m_1\}$ are identical distribution. If H fulfills the requirement, then $\{t \leftarrow \{0, 1\}^\lambda : H(t)\}$ is the uniform distribution over $\{0, 1\}^\lambda$. Consequently, the two distributions are identical. This thus concludes the proof that the hashed scheme is semantically secure. Thus, PEC2P is CPA secure. □

5.3. Efficiency Analysis. For a reporting leaf node, the computational cost only consists of one hash computation and one modular addition. For an aggregator, the computational cost consists of the sum operation of count and sum of perturbed data. If an aggregator has reporting data, it also has one hash computation.

Table 3: Number of bits sent per node for leaf node.

Protocol	Number of bits
O-ASP [All]	$\|h\| + 2 * \|\text{Per}\|$
O-ASP [Non]	$\|h\| + 2 * \|\text{Per}\| + \|\text{ID}\|$
Claude_09	$\|h\| + \|\text{Per}\|$
PEC2P	$h + \log_2 N + \|\text{Per}\|$

ID: node ID; h: header; Per: perturbed data; N: number of nodes in network.

We assume that there are N sensor nodes in reporting area and aggregation tree has a branching factor d of 3. Perturbed data Per = header + data + append. We choose the packet format used in TinyOS [16], and the packet header is 56 bits. Data is in the range of $[0, 127]$. Let count length, ID length, and append length be $\log_2 N$ bits. We consider two different scenarios: (1) only nodes at the lowest level may have data satisfying *BS*'s query and (2) nodes at each level may have data satisfying *BS*' query.

O-ASP [9] is designed based on an ideal and unrealistic assumption that each sensor node knows the membership and topology of the whole network and it knows whether each of these nodes has data satisfying each particular query. In each aggregation, a decision node (say *BS*) first compares the communication cost of [All-reporting] (A) and [Non-redundant-reporting] (N) for each cell and then decides which strategy will be chosen.

In Claude_09 [11], in the data aggregation phase, for scenario (1), each reporting node sends ($|\text{Per}|$) bits of message to its parent node, and nodes at second lowest level decide which group if IDs to send: the reporting nodes' IDs or the nonreporting nodes' IDs. For scenario (2), each reporting node will send ($|\text{ID}| + |\text{Per}|$) bits of message.

For PEC2P, in the data aggregation phase, for scenarios (1) and (2), each reporting node sends ($|\text{count}| + |\text{Per}|$) bits of message to its parent node, and the same length of message will also be sent from aggregators. No ID is transmitted in the aggregation tree.

We show the number of bits sent by leaf node in Table 3. Then, we calculate the average/maximum/minimum

Table 4: Theoretical analysis of total communication overhead.

(a)

	Claude_09
Average	$\sum_{i=\lceil \log_d N \rceil - 1}^{1} \left(\left(1 - (1-P)^{\sum_{j=1}^{\lceil \log_d N \rceil - i} d^j} \right) * \lvert H \rvert * d^i \right) + P * \lvert H \rvert * d^{\lceil \log_d N \rceil} + P' * \lvert \mathrm{ID} \rvert * d^{\lfloor \log_d N \rfloor} * \left(\lceil \log_d N \rceil - 1 \right)$
Minimum	$\lvert H \rvert * n + \lvert \mathrm{ID} \rvert * \left(\sum_{i=1}^{\lfloor \log_d n \rfloor} i * d^i + \lfloor \log_d n \rfloor * \left(n - d^{\lfloor \log_d n \rfloor} \right) \right)$
Maximum	
$C1$	$\lvert H \rvert * \left(\sum_{i=1}^{\lfloor \log_d n \rfloor} d^i + \left(n - d^{\lfloor \log_d n \rfloor} \right) * \left(\lfloor \log_d N \rfloor - \lfloor \log_d n \rfloor \right) + \lvert \mathrm{ID} \rvert * n * \lfloor \log_d N \rfloor \right)$
$C2$	$\lvert H \rvert * \left(N - d^{\lfloor \log_d N \rfloor} + n \right) + \lvert \mathrm{ID} \rvert * n * \lfloor \log_d N \rfloor$
$C3$	$\lvert H \rvert * N + \lvert \mathrm{ID} \rvert * \left(\sum_{j=\log_d \lceil \log_d N \rceil}^{i} j * d^j + (i-1) * n - S^i \right)$
	PEC2P
Average	$\sum_{i=\lceil \log_d N \rceil - 1}^{1} \left(1 - (1-P)^{\sum_{j=1}^{\lceil \log_d N \rceil - i} d^j} \right) * \lvert m \rvert * d^i + P * \lvert m \rvert * d^{\lceil \log_d N \rceil}$
Minimum	$\lvert m \rvert * n$
Maximum	
$C1$	$\lvert m \rvert * \left(\sum_{i=1}^{\lfloor \log_d n \rfloor} d^i + \left(n - d^{\lfloor \log_d n \rfloor} \right) * \left(\lceil \log_d N \rceil - \lfloor \log_d n \rfloor \right) \right)$
$C2$	$\lvert m \rvert * \left(N - d^{\lceil \log_d N \rceil} + n \right)$
$C3$	$\lvert m \rvert * N$

Note: only nodes at the lowest level may have data satisfying *BS*'s query.
$n = P * N$; d: degree; N: number of nodes in network; and $P' = P$ if $P \leq 0.5$ or $P' = 1 - P$.
ID: node ID; H = header + data + appendedBit; m = count + header + data + appendedBit.
C_1: $n < d^{\lfloor \log_d N \rfloor - 1}$; C_2: $d^{\lfloor \log_d N \rfloor - 1} < n < d^{\lfloor \log_d N \rfloor}$; C_3: $n > d^{\lfloor \log_d N \rfloor}$; $S_j = \sum_{\log_d \lceil N \rceil}^{j} d^j$, and $S^i < n < S^{i-1}$.

(b)

	Claude_09
Average	$\sum_{i=\lceil \log_d N \rceil - 1}^{1} \left(P * \lvert \mathrm{ID} \rvert * d^i * (i-1) + \left(1 - (1-P)^{\sum_{j=0}^{\lceil \log_d N \rceil - i} d^j} * \lvert H \rvert * d^{i-1} \right) \right) + P * d^{\lceil \log_d N \rceil} * \lvert \mathrm{ID} \rvert * \left(\lceil \log_d N \rceil - 1 \right) + P * \lvert H \rvert * d^{\lceil \log_d N \rceil}$
Minimum	$\lvert H \rvert * n + \lvert \mathrm{ID} \rvert * \left(\sum_{i=1}^{\lfloor \log_d n \rfloor} i * d^i + \left(\lfloor \log_d n \rfloor + 1 \right) * \left(n - d^{\lfloor \log_d n \rfloor} \right) \right)$
Maximum	
$C1$	$\lvert H \rvert * \left(\sum_{i=1}^{\lfloor \log_d n \rfloor} d^i + \left(n - d^{\lfloor \log_d n \rfloor} \right) * \left(\lfloor \log_d N \rfloor - \left(\lfloor \log_d n \rfloor \right) \right) + \lvert \mathrm{ID} \rvert * n * \lfloor \log_d N \rfloor \right)$
$C2$	$\lvert H \rvert * \left(N - d^{\lfloor \log_d N \rfloor} + n \right) + \lvert \mathrm{ID} \rvert * n * \lfloor \log_d N \rfloor$
$C3$	$\lvert H \rvert * N + \lvert \mathrm{ID} \rvert * \left(\sum_{j=\log_d \lceil \log_d N \rceil}^{i} j * d^j + (i-1) * n - S^i \right)$
	PEC2P
Average	$\sum_{i=\lceil \log_d N \rceil - 1}^{1} \left(1 - (1-P)^{\sum_{j=0}^{\lceil \log_d N \rceil - i} d^j) * \lvert m \rvert * d^{i-1}} + P * \lvert m \rvert * d^{\lceil \log_d N \rceil} \right)$
Minimum	$\lvert m \rvert * n$
Maximum	
$C1$	$\lvert m \rvert * \left(\sum_{i=1}^{\lfloor \log_d n \rfloor} d^i + \left(n - d^{\lfloor \log_d n \rfloor} \right) * \left(\lceil \log_d N \rceil - \lfloor \log_d n \rfloor \right) \right)$
$C2$	$\lvert m \rvert * \left(N - d^{\lceil \log_d N \rceil} + n \right)$
$C3$	$\lvert m \rvert * N$

Note: nodes at each level may have data satisfying *BS*'s query.
$n = P * N$; d: degree; N: number of nodes in network; and $P' = P$ if $P \leq 0.5$ or $P' = 1 - P$.
ID: node ID; H = header + data + appendedBit; m = count + header + data + appendedBit.
C_1: $n < d^{\lfloor \log_d N \rfloor - 1}$; C_2: $d^{\lfloor \log_d N \rfloor - 1} < n < d^{\lfloor \log_d N \rfloor}$; C_3: $n > d^{\lfloor \log_d N \rfloor}$; $S_j = \sum_{\log_d \lceil N \rceil}^{j} d^j$, and $S^i < n < S^{i-1}$.

Table 5: Number of bits sent per node for each level with Claude_09 scheme.

Level	Number node	A (100%)	A (90%)	A (70%)	AV (100%)	AV (90%)	AV (70%)	HBH-A	HBH-AV	No-Agg
1	3	75	949.8	2699.4	100	974.8	2724.4	73	97	68859
2	9	75	366.6	949.8	100	391.6	974.8	72	94	22932
3	27	75	172.2	366.6	100	197.2	391.6	70	91	7623
4	81	75	107.4	172.2	100	132.4	197.2	68	87	2520
5	243	75	85.8	107.4	100	110.8	132.4	67	84	819
6	729	75	78.5	83.8	100	103.5	108.1	65	81	252
7	2187	75	67.5	52.5	100	90	70	63	63	63

Note: only the nodes in the lowest level may have data satisfying *BS*'s query.

Table 6: Number of bits sent per node for each level with PEC2P scheme.

Level	Number Node	A (100%)	A (90%)	A (70%)	AV (100%)	AV (90%)	AV (70%)	HBH-A	HBH-AV	No-Agg
1	3	87	87	87	112	112	112	73	97	68859
2	9	87	87	87	112	112	112	72	94	22932
3	27	87	87	87	112	112	112	70	91	7623
4	81	87	87	87	112	112	112	68	87	2520
5	243	87	87	87	112	112	112	67	84	819
6	729	87	86.9	84.7	112	111.9	109	65	81	252
7	2187	87	78.3	60.9	112	100.8	78.4	63	63	63

Note: only nodes in the lowest level may have data satisfying *BS*'s query.

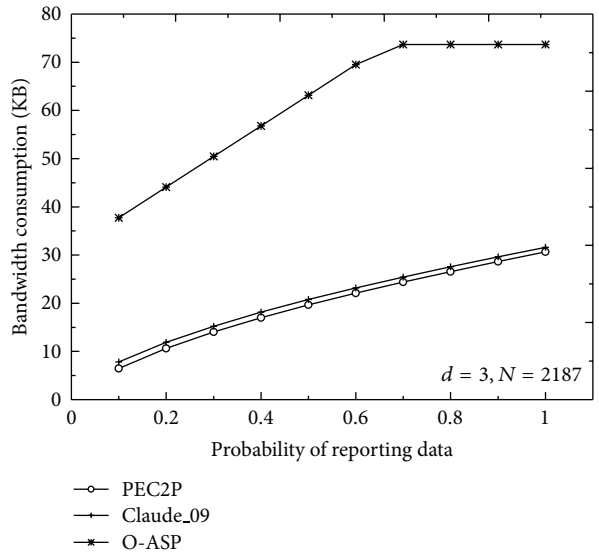


Figure 3: Communication overhead with different probability of reporting data when only nodes in the lowest level may have data satisfying *BS*' query.

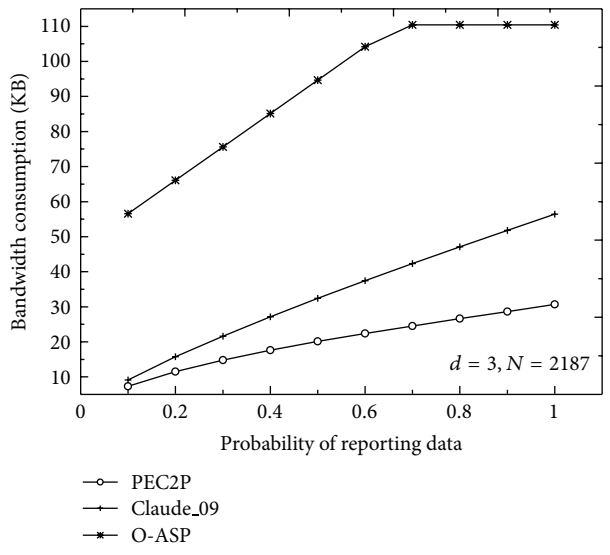


Figure 4: Communication overhead with different probability of reporting data when nodes at each level may have data satisfying *BS*' query.

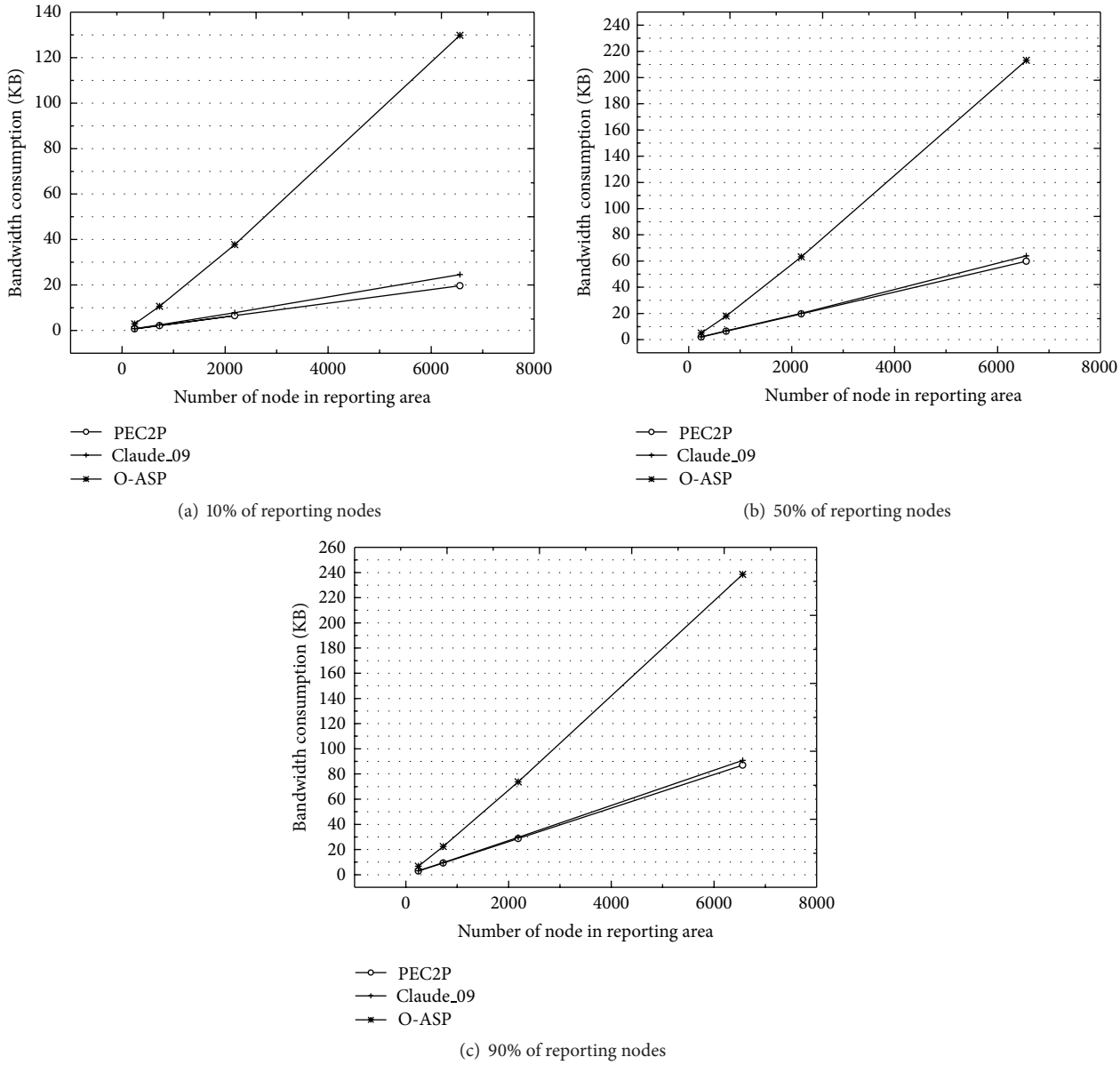


Figure 5: Bandwidth consumption in data aggregation phase when only nodes in the lowest level may have data satisfying *BS*' query.

communication overhead CO in aggregation phase for Claude_09 and PEC2P in Table 4. In minimum case, reporting nodes are located in the high levels of aggregation tree, and we can find them through breadth-first search. In maximum case, reporting nodes should be located from the lowest level to higher levels. Tables 5 and 6 list the number of bits sent per node for each level with Claude_09 and PEC2P.

Figures 3 and 4 show the trend of communication overhead in two different scenarios.

We assume that only the nodes in the lowest level have a probability of $P(= 0.1, 0.5, 0.9)$ to sense environmental data. Results are shown in Figures 5(a), 5(b), and 5(c).

We further assume that all nodes in aggregation tree has a probability of $P(= 0.1, 0.5, 0.9)$ to sense environmental data. Results are shown in Figures 6(a), 6(b), and 6(c).

Results show that, compared with existing protocols, PEC2P can greatly reduce communication overhead in aggregation phase. We notice that the major communication overhead is caused by transferring the hash value which was computed by SHA-1 in the comparison. Performance can be further optimized by choosing other hash functions with shorter output in case of lower security level requirement.

Result Retrieving Algorithm Test. We used a computer with a Pentium(R) D CPU of 3.40 GHZ and 2.00 GB memory to test Algorithm 7. Since sensor nodes are relatively uniformly distributed and their communication range is from 50 meters to 100 meters, a local event will be detected by a small group of sensor nodes. Therefore, we choose to use a small N. Results show that choosing 5 nodes from 10 nodes only needs 8 milliseconds and choosing 10 nodes from 20 nodes only needs

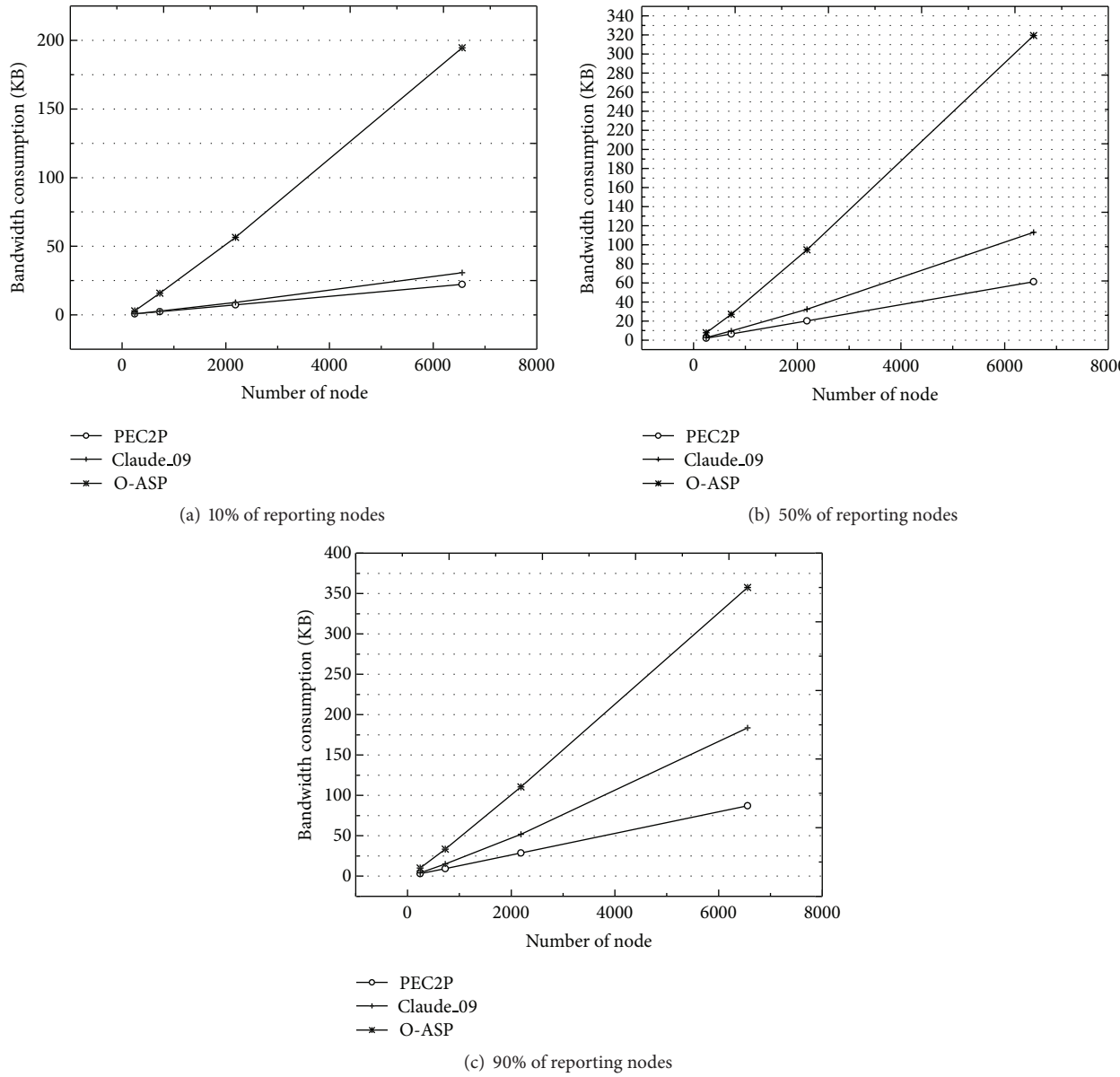


Figure 6: bandwidth consumption in data aggregation phase when nodes at each level may have data satisfying *BS*' query.

approximately 2 seconds. In WSNs, the capability of *BS* is more powerful than our experimental computer; thus, the searching time will be shorter in real applications. To make the search efficiently, we can first divide the network into clusters of trees.

6. Conclusion

Confidentiality protection and energy efficiency are two conflict, but equally crucial requirements in WSNs. To achieve a trade-off between these two goals simultaneously, remains a challenge. We propose PEC2P to protect data confidentiality which also achieves energy efficiency. Specifically, we need no ID list and use one-way hash function as perturbation added to the environmental data. Since *BS* usually has powerful computation capacities, we utilize *BS* to the fullest and let it compute which nodes have actually contributed to the aggregation process after receiving the final perturbed aggregation result. Consequently, we manage to preserve data confidentiality, avoid high energy consumption, and obtain lower overall communication overhead. Analysis and experiments have also been conducted to evaluate the proposed protocol. The results show that our protocol provides confidentiality protection for both raw and aggregated data with an overhead lower than that of the existing related protocols. PEC2P can be adopted to tree/cluster-based aggregation and any protocol using ID-list transmission. We focus on collecting the number of contributing nodes and its perturbed data, instead of how the information is gathered. For uniformity,

```
begin
  Agg ← −1; i ← 0; c ← 0;
  for i ← 0 to C − 1 do
    IDL[i] ← 1;
  for i: C to N − 1 do
  IDL[i] ← 0;
  Agg ← Matching(IDL, HAX_BS);
  if Agg ≠ − 1 then
    return ⟨IDL, Agg⟩;
  /*search C_N^C − 1 times*/
  for order ← 1 to C_N^C − 1 do
    for i ← 0 to N − 1 do
      if IDL[i] = 1 then
        c ← c + 1;
        /*find the last '1' in IDL[ ]*/
        if c = C and i ≤ N − 1 then
          IDL[i + 1] ← 1; IDL[i] ← 0;
          Agg ←Matching(IDL, HAX_BS);
          if Agg ≠ − 1 then
            return ⟨IDL, Agg⟩;
          i ← N; c ← 0;
        /*The last '1' is in the last position, then
        move the last continuous '1's to the first
        found '1' before them*/
        if c = C and i = N − 1 then
          /how many '1's should be moved*/
          group ← 0;
          /*from behind*/
          for j ← N − 1 to 0 do
            if IDL[j] = 1 then
              group ← group + 1;
            /*found the empty position and
            move the newly found '1' and the
            continuous '1's*/
            else
              /*newly found '1'*/
              m ← 0;
              /continuous '1's' new location*/
              NewLocation ← 0;
              for m ← j − 1 to 0 do
                if IDL[m] = 1 then
                  IDL[m] ← 0;
                  IDL[m + 1] ← 1;
                  NewLocation ← m + 2;
                  /*searching ends*/
                  m ← −1;
            /*move the continuous '1's*/
            if NewLocation < N then
              for p ← N − 1 to N − group
              do
                IDL[p] ← 0;
              for q ← NewLocation to
              group + NewLocation − 1 do
                IDL[q] ← 1;
            j ← −1;
        Agg ←Matching(IDL, HAX_BS);
        if Agg ≠ − 1 then
          return ⟨IDL, Agg⟩;
          i ← N; c ← 0;
end
```

Algorithm 7: Result retrieving algorithm.

we use tree topology in our paper. We also did cluster-based comparison with existing protocols, and the results show no significant difference.

Appendix

For more details, see Algorithms 6 and 7.

Acknowledgments

This paper is supported by the National Natural Science Foundation of China (nos. 61272512, 61003262, and 61100172), Program for New Century Excellent Talents in University (NCET-12-0047), and Beijing Natural Science Foundation (no. 4121001).

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[3] M. Li and Y. Liu, "Iso-Map: energy-efficient contour mapping in wireless sensor networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 5, pp. 699–710, 2010.

[4] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tag: a tiny aggregation service for ad-hoc sensor networks," in *Proceedings of the 5th Symposium on Operating Systems Design and Implementation ACM SIGOPS Operating Systems Review (OSDI '02)*, vol. 36, pp. 131–146, 2002.

[5] K. Akkaya, M. Demirbas, and R. S. Aygun, "The impact of data aggregation on the performance of wireless sensor networks," *Wireless Communications and Mobile Computing*, vol. 8, no. 2, pp. 171–193, 2008.

[6] C. Castelluccia, E. Mykletun, and G. Tsudik, "Efficient aggregation of encrypted data in wireless sensor networks," in *Proceedings of the 2nd Annual International Conference on Mobile and Ubiquitous Systems—Networking and Services (MobiQuitous '05)*, pp. 109–117, July 2005.

[7] J. Girao, D. Westhoff, and M. Schneider, "CDA: concealed data aggregation for reverse multicast traffic in wireless sensor networks," in *Proceedings of IEEE International Conference on Communications (ICC '05)*, pp. 3044–3049, May 2005.

[8] E. Mykletun, J. Girao, and D. Westhoff, "Public key based cryptoschemes for data concealment in wireless sensor networks," in *Proceedings of the 41st IEEE International Conference on Communications (ICC '06)*, pp. 2288–2295, July 2006.

[9] T. Feng, C. Wang, W. Zhang, and L. Ruan, "Condentiality protection for distributed sensor data aggregation," in *Proceedings of the 27th IEEE International Conference on Computer Communications (INFOCOM '08)*, pp. 68–76, 2008.

[10] J. Albath and S. Madria, "Secure hierarchical data aggregation in wireless sensor networks," in *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC '09)*, pp. 1–6, April 2009.

[11] C. Castelluccia, A. C. F. Chan, E. Mykletun, and G. Tsudik, "Efficient and provably secure aggregation of encrypted data in wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 5, no. 3, pp. 1–36, 2009.

[12] S. Ozdemir and Y. Xiao, "Secure data aggregation in wireless sensor networks: a comprehensive overview," *Computer Networks*, vol. 53, no. 12, pp. 2022–2037, 2009.

[13] L. Hu and D. Evans, "Secure aggregation for wireless networks," in *Proceedings of the 3rd IEEE Symposium on Applications and the Internet Workshops (SAINT '03)*, p. 384, IEEE Computer Society, Washington, DC, USA.

[14] A. Mahimkar and T. S. Rappaport, "SecureDAV: a secure data aggregation and verification protocol for sensor networks," in *Proceedings of the 47th IEEE Global Telecommunications Conference (GLOBECOM'04)*, pp. 2175–2179, December 2004.

[15] S. Ozdemir and Y. Xiao, "Polynomial regression based secure data aggregation for wireless sensor networks," in *Proceedings of the 54th IEEE Global Telecommunications Conference (GLOBECOM '11)*, pp. 1–5, 2011.

[16] C. Karlof, N. Sastry, and D. Wagner, "TinySec: a link layer security architecture for wireless sensor networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*, pp. 162–175, November 2004.

[17] J. Katz and A. Y. Lindell, *Modern Crytography*, Chapman & Hall, 2008.

[18] 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, November 2003.

[19] J. M. McCune, E. Shi, A. Perrig, and M. K. Reiter, "Detection of denial-of-message attacks on sensor network broadcasts," in *Proceedings of the 25th IEEE Symposium on Security and Privacy (SP '05)*, pp. 64–78, May 2005.

[20] A. D. Wood and J. A. Stankovic, "Denial of service in sensor networks," *Computer*, vol. 35, no. 10, pp. 54–62, 2002.

[21] J. Newsome, E. Shi, D. Song, and A. Perrig, "The Sybil attack in sensor networks: analysis and defenses," in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN '04)*, pp. 259–268, April 2004.

[22] B. Parno, A. Perrig, and V. Gligor, "Distributed detection of node replication attacks in sensor networks," in *Proceedings of the 25th IEEE Symposium on Security and Privacy (SP '05)*, pp. 49–63, May 2005.

[23] S. Brands and D. Chaum, "Distance-bounding protocols," in *Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques on Advances in Cryptology (EUROCRYPT '93)*, vol. 839 of *Lecture Notes in Computer Science*, pp. 344–359, Springer, 1994.