Sensors **2014**, *14*, 6701-6721; doi:10.3390/s140406701

OPEN ACCESS

sensors

ISSN 1424-8220

www.mdpi.com/journal/sensors

Article

A Secure-Enhanced Data Aggregation Based on ECC in Wireless Sensor Networks

Qiang Zhou [1,2,3,*], Geng Yang [2,*] and Liwen He [1]

[1] Institute of Advanced Technology, Nanjing University of Posts & Telecommunications, Nanjing 210046, China; E-Mail: helw@njupt.edu.cn

[2] School of Computer Science & Technology, Nanjing University of Posts & Telecommunications, Nanjing 210046, China

[3] School of Computer & Information Engineering, Chuzhou University, Chuzhou 239000, China

***** Authors to whom correspondence should be addressed;
E-Mails: zhouqiang@njupt.edu.cn (Q.Z.); yangg@njupt.edu.cn (G.Y.);
Tel.: +86-150-5055-9266 (Q.Z.).

Received: 16 December 2013; in revised form: 24 March 2014 / Accepted: 8 April 2014 / Published: 11 April 2014

Abstract: Data aggregation is an important technique for reducing the energy consumption of sensor nodes in wireless sensor networks (WSNs). However, compromised aggregators may forge false values as the aggregated results of their child nodes in order to conduct stealthy attacks or steal other nodes' privacy. This paper proposes a Secure-Enhanced Data Aggregation based on Elliptic Curve Cryptography (SEDA-ECC). The design of SEDA-ECC is based on the principles of privacy homomorphic encryption (PH) and divide-and-conquer. An aggregation tree disjoint method is first adopted to divide the tree into three subtrees of similar sizes, and a PH-based aggregation is performed in each subtree to generate an aggregated subtree result. Then the forged result can be identified by the base station (BS) by comparing the aggregated count value. Finally, the aggregated result can be calculated by the BS according to the remaining results that have not been forged. Extensive analysis and simulations show that SEDA-ECC can achieve the highest security level on the aggregated result with appropriate energy consumption compared with other asymmetric schemes.

Keywords: wireless sensor networks; data aggregation; Elliptic Curve Cryptography (ECC); data integrity; data privacy

1. Introduction

Wireless sensor networks (WSNs) consist of thousands of sensors that collect data from a certain deployed range. Currently, WSNs have plenty of applications, such as military investigation, environment monitoring and accident reporting, *etc.* Typically, sensors have strictly limited computation and communication abilities and power resources; therefore, reducing the power consumption is a critical concern for WSNs. For better energy utilization, data aggregation [1,2] has been proposed recently. The original concept is to aggregate multiple sensing messages by performing statistical or algebraic operations, such as addition, minimum, maximum, median, *etc.* Since only the aggregated results need to reach the base station (BS) instead of sensing data, communication costs can be significantly reduced. Unfortunately, data aggregation is vulnerable to some attacks. For example, an adversary could compromise cluster heads (aggregators) similar to compromising all its cluster members. To solve this problem, several schemes, such as SDAP [3], PEPDA [4], Jung *et al.*'s scheme [5] have been proposed. However, these schemes can only guarantee the data privacy during the process of data aggregation and have a long aggregation delay.

An alternative method for secure data aggregation is to use privacy homomorphic encryption (PH), which can aggregate encrypted messages directly from sensors without decrypting so that it has a short aggregation delay. An adversary knows nothing from forging aggregated results even if the aggregators are compromised, because aggregators are unable to encrypt messages. PH is allowed to carry out specific types of computations on ciphertext, and the decrypted aggregation result matches the result of operations performed on the plaintext. PH has been used for data aggregation in WSNs, such as in Wang *et al.*'s scheme [6], CDAMA [7], Tiny PEDS [8], *etc.* However, the existing PH schemes suffer from the data integrity issue.

In this paper, we focus on bridging the gap between data privacy and integrity in WSNs. Some symmetric secure aggregation schemes [9,10] have been proposed to achieve both data privacy and integrity, but they cannot defend against node compromise attacks due to its inherent drawback that the encryption key is same as the decryption key. In general, symmetric schemes are less secure than asymmetric ones, although they are more efficient in terms of computational cost. Therefore, we originally propose a secure-enhanced data aggregation scheme based on Elliptic Curve Cryptography (ECC), called SEDA-ECC, which is an improved version of Boneh *et al.*'s asymmetric scheme [11]. To the best of our knowledge, SEDA-ECC can defend against the most attacks with appropriate energy consumption compared with other asymmetric schemes.

The rest of the paper is organized as follows: in Section 2, the existing secure data aggregation schemes in WSNs are presented. The system model and preliminaries are discussed in Section 3. In Section 4, a secure-enhanced data aggregation scheme based on ECC is proposed. Section 5 describes the security analysis of SEDA-ECC, and Section 6 presents performance evaluation and comparison to prove the effectiveness and efficiency of our scheme. Finally, we conclude SEDA-ECC in Section 7.

2. Related Works

Currently, many secure data aggregation schemes have been proposed. For symmetric schemes, Ozdemir *et al.* [9] integrated false data detection with data aggregation and confidentiality, and proposed an authentication protocol. In the scheme, every aggregator has some monitoring nodes which also perform data aggregation for data verification, and the integrity of the encrypted data is verified by the sensors between two consecutive aggregators. Its limitation is the rigorous topological constraints. Papadopoulos *et al.* [10] presented an exact aggregation scheme with integrity and confidentiality, named SIES. SIES combines the symmetric homomorphic encryption with secret sharing. A wide range of aggregates can be covered, and a small amount of bandwidth consumption is introduced in SIES. However, the data transmission efficiency is low due to the oversize space of secret keys. Based on Aggregation-Commit-Verify approach, Chan *et al.* [12] first proposed a provably secure hierarchical data aggregation scheme, where the adversary is forced to commit to its choice of aggregation results, then the sensors are allowed to verify whether their aggregation contributions are correct or not. The scheme can be used for multiple malicious nodes and arbitrary topologies, but it inherits the weakness of large amount of communication and computation overheads. To address this issue, Frikken *et al.* [13] improve Chan's scheme by reducing the maximum communication per node from $O(\Delta\log^2 n)$ to $O(\Delta\log n)$, where n is the number of nodes in WSNs, and Δ is the maximum degree of the aggregation tree.

For asymmetric schemes, Zhu *et al.* [14] focused on preserving data integrity and proposed an efficient integrity-preserving data aggregation protocol named EIPDAP. The scheme is based on the modulo addition operation using ECC, and has the most optimal upper bound on solving the integrity-preserving problem for data aggregation. Niu *et al.* [15] proposed a secure identity-based lossy data aggregation scheme using homomorphic hashing and identity-based aggregate signature. In the scheme, the authenticity of aggregated data can be verified by both aggregators and BS. The computation and communication overheads could be significantly reduced because the BS can perform batch verification. However, the above two schemes may lead to the leakage of data privacy due to decryption at the aggregator. Based on PH, Westhoff *et al.* [16] and Girao *et al.* [17] proposed CDA methods to facilitate aggregation in encrypted data, where richer algebraic operations can be directly executed on encrypted data by aggregators. Mykletun *et al.* [18] adopted several public-key-based PH encryptions to achieve data concealment in WSNs. Furthermore, Girao *et al.* [8] proposed a novel scheme by extending the ELGamal PH encryption. However, the above schemes cannot resist node compromise attacks. Specific security analysis is presented in Section 5.

3. System Model and Preliminaries

In this section, we describe the aggregation model and the attack model. The aggregation model defines how aggregation works, and the attack model defines what kinds of attacks our secure data aggregation scheme should protect against.

3.1. Aggregation Model

We consider large scale WSNs with densely deployed sensors. In WSNs, there are three types of nodes: base station (BS), aggregator, and leaf node. In this paper, we consider the aggregation tree roots at the BS like general data aggregation protocol [1,3]. Sensor nodes have overlapping sensing regions due to the dense deployment, and the same event is often detected by multiple sensors. Hence, data aggregation is proposed to reduce data transmission. The non-leaf nodes, except the BS, may also serve as aggregators. They are responsible for combining answers from their child nodes and forwarding intermediate aggregation results to their parents. Without loss of generality, we focus on additive aggregation, which can serve as the base of other statistical operations (e.g., count, mean, or variance).

3.2. Attack Model

First, we categorize the abilities of the adversary as follows:

(1) An adversary can eavesdrop on transmission data in a WSN.
(2) An adversary can send the forged data to leaf nodes, aggregators, or BS.
(3) An adversary can compromise secrets in sensors or aggregators.

Then, we define five attacks to qualify the security strength of the secure data aggregation schemes, based on adversary's abilities and purposes.

(1) Ciphertext analysis

Ciphertext analysis is a very common and basic attack. In such an attack, an adversary wants to deduce the secret key or obtain information only by interpreting ciphertext. A secure scheme must ensure that it is not possible to gain any information or key, and an adversary cannot decide whether an encrypted ciphertext corresponds to a specific plaintext or not.

(2) Chosen plaintext attacks

Given some chosen samples of plaintexts and corresponding ciphertexts, the adversary can determine secret information or deduce the key. A secure scheme must ensure that an adversary cannot deduce secret keys or additional information out of the known set, even with a large set of plaintexts and their ciphertexts.

(3) Malleability

The aim of the adversary is to alter the valid ciphertexts without leaving marks. In this kind of attack, an attack can randomly generate meaningless ciphertexts that are syntactically correct to harm the system. For many PH schemes, it is possible to alter the ciphertexts without knowing the concrete content. Hence, a secure scheme should not let the adversary be able to successfully change the contents of encrypted packet.

(4) Unauthorized aggregation

In this kind of attack, an adversary is to aggregate two or more ciphertexts into forged but format-valid ciphertexts, then to inject them into the network for vandalizing the system.

(5) Node compromise attacks

An adversary can compromise sensors or aggregators. When an adversary compromises an aggregator and gets its secret, it can easily launch unauthorized aggregation and malleability attacks. When an adversary compromises a sensor and gets its secret, it can decrypt the ciphertexts of all sensors in the symmetric schemes; besides, it also can impersonate the sensor or the other sensors to generate legal ciphertexts in both symmetric and asymmetric schemes.

3.3. Privacy Homomorphism

A privacy homomorphism is an encryption transformation which allows direct computation on the encrypted data. Let m_1 and m_2 be two plaintexts, and $\otimes$, $\times$ be the homomorphic operations on the ciphertexts and plaintexts respectively, we have Enc(m_1) $\otimes$ Enc(m_2) = Enc($m_1 \times m_2$), where Enc(m) represents the ciphertext of m. Component-wise multiplications and additions of ciphertexts result in the corresponding multiplications and additions of plaintexts. If $E_{(p,q)}(m_1) = (x_1,y_1)$ and $E_{(p,q)}(m_2) = (x_2,y_2)$, then:

$$\begin{aligned} E_{(p,q)}(m_1+m_2) &= Add(E_{(p,q)}(m_1), E_{(p,q)}(m_2)) \\ &= (x_1+x_2 \bmod n, y_1+y_2 \bmod n) \end{aligned} \tag{1}$$

$$\begin{aligned} E_{(p,q)}(m_1 m_2) &= Multi(E_{(p,q)}(m_1), E_{(p,q)}(m_2)) \\ &= (x_1 x_2 \bmod n, y_1 y_2 \bmod n) \end{aligned} \tag{2}$$

However, symmetric cryptography-based privacy homomorphism has been proved to be insecure in chosen plaintext attacks for some specific parameters [19]. Therefore, privacy homomorphism based on asymmetric cryptography should be used instead of privacy homomorphism based on symmetric cryptography for some mission critical networks.

3.4. BGN Scheme

Boneh *et al.* [11] propose a PH scheme (abbreviated as BGN) based on the encryption schemes proposed by Paillier [20] and Okamoto-Uchiyama [21]. Both additive and multiplicative homomorphisms are provided in BGN, however, multiplicative homomorphism is inefficient and very expensive for WSNs because it is based on the bilinear pairing. Hence, we only adapt additive homomorphism of BGN to our scheme. The additive homomorphic encryption of BGN can be applied to private data aggregation, which is described in Algorithm 1.

Due to large computational overhead of the asymmetric cryptography, Boneh *et al.* construct BGN on a cyclic group of elliptic curve point. In phase 1 of BGN scheme, supposing E is the set of elliptic curve points that form a cyclic group, ord(E) denotes the number of points in E. Supposing $\mathcal{G}$ is a point in E, ord($\mathcal{G}$) denotes the order of a point $\mathcal{G}$. If ord($\mathcal{G}$) = q, there is $q*\mathcal{G} = \infty$, where ∞ is the identity element of the group. In phase 2, point addition and scalar multiplication over points $\mathcal{G}$ and $\mathcal{H}$ are used to encrypt the message M. Ciphertext C is composed of the message part and the secure randomness. In phase 3, BGN can aggregate the ciphertext due to homomorphic property. As we can see, the aggregated result will be the form of $\sum M*\mathcal{G} + \sum R*\mathcal{H}$, where $\sum M$ is the sum of the messages, and $\sum R$ is the sum of the randomness. In phase 4, BGN can decrypt the aggregated result to get the plaintext by multiplying the result with private key. When randomness of point $\mathcal{H}$ is removed by multiplying the order of $\mathcal{H}$, we can obtain ord($\mathcal{H}$)$*\sum M*\mathcal{G}$. Finally, the plaintext $\sum M$ can be retrieved by applying the discrete logarithm.

Algorithm 1. BGN scheme.

```
Phase 1. Key-Gen($\lambda$): generate a public-private key pair.
01: Compute ($q_1$, $q_2$, $E$) using security parameter $\lambda$, where $E$ is the set of elliptic curve points that form a
    cyclic group. ord($E$)= $n$= $q_1q_2$, where $q_1$, $q_2$ are large primes, and the bit lengths of them are the same, i.e.,
    $|q_1| = |q_2|$.
02: Randomly select two generators, $\mathcal{G}$ and $\mathcal{U}$ such that ord($\mathcal{G}$) = ord($\mathcal{U}$) = $n$.
03: Compute point $\mathcal{H} = q_2 * \mathcal{U}$ such that ord($\mathcal{H}$) = $q_1$.
04: Select parameter $T < q_1$ as the maximum plaintext boundary.
05: Generate Public key $PK = (n, E, \mathcal{G}, \mathcal{H}, T)$ and Private key $SK = q_1$.
Phase 2. Enc($PK$, $M$): message encryption on $M$ by public key $PK$.
01: Check if the message space of a sensor node $M \in \{0, 1, \ldots, T\}$.
02: Random pick up $R \in \{0, 1, \ldots, n-1\}$.
03: Generate the ciphertext $C = M*\mathcal{G} + R*\mathcal{H}$.
04: Return $C$.
Phase 3. Agg($C_1$, $C_2$): message aggregation on two ciphertexts $C_1$ and $C_2$,
where $C_i = M_i*\mathcal{G} + R_i*\mathcal{H}$, $i$ = 1,2.
01: Randomly select $R' \in \{0, 1, \ldots, n-1\}$.
02: Compute the aggregated ciphertext
    $C' = C_1 + C_2 + R'*\mathcal{H} = (M_1 + M_2)*\mathcal{G} + (R_1 + R_2 + R')*\mathcal{H}$
03: Return $C'$.
Phase 4. Dec($SK$, $C$): message decryption on $C$ using private key $SK$.
01: Compute $M = \log_{\tilde{\mathcal{G}}}(q_1 * C) = \log_{\tilde{\mathcal{G}}}\big(q_1 * (M * \mathcal{G} + R * \mathcal{H})\big) = \log_{\tilde{\mathcal{G}}}(q_1 * M * \mathcal{G})$, where $\tilde{\mathcal{G}} = q_1*\mathcal{G}$.
02: Return $M$.
```

4. SEDA-ECC: A Secure-Enhanced Data Aggregation Based on ECC

In this section, we modify BGN to fit the SEDA-ECC scheme, so the security of BGN and SEDA-ECC are all based on the hardness assumption of subgroup decision problem. If we only provide the privacy protection of data aggregation, BGN can be used in SEDA-ECC directly, however, we also aim to ensure the data integrity, hence, different public-private key pairs and disjoint aggregation tree will be adopted. We first describe the details of SEDA-ECC scheme, which consists of six phases listed in Algorithm 2, then we present a case study of SEDA-ECC.

4.1. Key Generation Phase

Given a security parameter $\lambda \in \mathbb{Z}$, the tuple (q_1, q_2, q_3, E) is generated. E is the set of elliptic curve points that form a cyclic group, and ord(E) = $n = q_1q_2q_3$, where q_1, q_2, q_3 are large primes, and the bit lengths of them are the same. Then, randomly select three points ($\mathcal{G}_1$, $\mathcal{G}_2$, $\mathcal{G}_3$) from E, where the order of $\mathcal{G}_i$ is n, i = 1, 2, 3. Compute point $\mathcal{P} = q_2q_3*\mathcal{G}_1$, $\mathcal{Q} = q_1q_3*\mathcal{G}_2$, and point $\mathcal{H} = q_1q_2*\mathcal{G}_3$, such that the order of $\mathcal{P}$, $\mathcal{Q}$ and $\mathcal{H}$ is q_1, q_2, and q_3 respectively.

Algorithm 2. SEDA-ECC scheme.

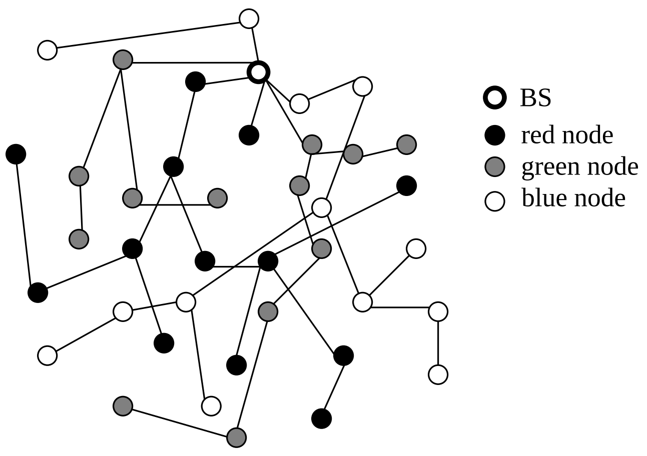
```
Input: An aggregated WSN and SQL type SUM aggregation query
Output: SUM aggregation result after integrity checked.
Phase 1. Key-Gen($\lambda$): generate public-private key pairs for tree $T_i$, where $i = r$, $g$ and $b$..
01: Compute ($q_{i1}, q_{i2}, q_{i3}, E$) based on security parameter $\lambda$, where $E$ is the set of elliptic curve points that form
    a cyclic group. ord($E$) = $n = q_{i1}q_{i2}q_{i3}$, where $q_{i1}, q_{i2}, q_{i3}$ are large primes, and the bit lengths of them are the
    same, i.e., $|q_{i1}| = |q_{i2}| = |q_{i3}|$.
02: Randomly select generators, $\mathcal{G}_{i1}, \mathcal{G}_{i2}$ and $\mathcal{G}_{i3}$ such that ord($\mathcal{G}_{i1}$) = ord($\mathcal{G}_{i2}$) = ord($\mathcal{G}_{i3}$) = $n$.
03: Compute point $\mathcal{H}_i = q_{i1}q_{i2} * \mathcal{G}_{i3}$ such that ord($\mathcal{H}_i$) = $q_{i3}$, $\mathcal{P}_i = q_{i2}q_{i3} * \mathcal{G}_{i1}$ such that ord($\mathcal{P}_i$) = $q_{i1}$, and
    $\mathcal{Q}_i = q_{i1}q_{i3} * \mathcal{G}_{i2}$ such that ord($\mathcal{Q}_i$) = $q_{i2}$. Then output Public key $PK = (n, E, \mathcal{P}_i, \mathcal{Q}_i, \mathcal{H}_i)$ and Private key
    $SK = \{(q_{i1}q_{i3}), (q_{i2}q_{i3})\}$.
Phase 2. Dis-Tree($p_r, p_g$, $\mathrm{p}_b$): disjoint aggregation tree construction with probability $p_r$, $p_g$ and $\mathrm{p}_b$.
01: BS triggers the aggregation by a HELLO message, when receiving such a message, nodes select their roles:
    red aggregator, green aggregator and blue aggregator. Aggregators then also forward the HELLO messages.
02: If a node receives HELLO messages from red, green and blue aggregators, it randomly selects its role
    according to $p_i$; otherwise it waits until the HELLO messages from all kinds of aggregators are received.
03: Three disjoint aggregation trees rooted at the BS can be formed as the disjoint tree construction procedure
    continues. Red aggregators, green aggregators and blue aggregators interleave with each other.
Phase 3. Enc($PK, M_i$): message encryption in three trees respectively, where $i = r$, $g$ and $b$.
01: Set $T_M < q_{i1}$. Check if the message space of a sensor node $M_i \in \{0, 1, \ldots, T_M\}$.
02: Randomly pick up $R_i \in \{0, 1, \ldots, n - 1\}$.
03: Generate the ciphertext $C_i = M_i * \mathcal{P}_i + \mathcal{Q}_i + R_i * \mathcal{H}_i$.
04: Return $C_i$.
Phase 4. Agg($C_{i1}, C_{i2}$): message aggregation on two ciphertexts $C_{i1}$ and $C_{i2}$, where $i = r$, $g$ and $b$.
01: Compute the aggregated ciphertext
    $C_{ia} = C_{i1} + C_{i2} = (\sum M_{ij}) * \mathcal{P}_i + \zeta_i * \mathcal{Q}_i + (\sum R_{ij}) * \mathcal{H}_i$,
    where $\sum M_{ij}$ represents the aggregated result of tree $T_i$, $\zeta_i$ represents the number of aggregated ciphertexts in
    tree $T_i$, and $\sum R_{ij}$ represents the aggregated randomness in tree $T_i$.
02: Return $C_{ia}$.

Phase 5. Dec($SK, C_{ia}$): message decryption on $C_{ia}$ in tree $T_i$.

01: Compute $M_i = \sum M_{ij} = \log_{\widetilde{\mathcal{P}_i}}(q_{i2}q_{i3} * C_i)$, where $\widetilde{\mathcal{P}_i} = q_{i2}q_{i3} * \mathcal{P}_i$.

02: Compute $\zeta_i = \log_{\widetilde{\mathcal{Q}_i}}(q_{i1}q_{i3} * C_i)$, where $\widetilde{\mathcal{Q}_i} = q_{i1}q_{i3} * \mathcal{Q}_i$.
03: Return $M_i, \zeta_i$.
Phase 6. Chec($M_i$): check message $M_i$ integrity at the BS, where $i = r$, $g$ and $b$.
01: Set $i, j, k \in \{r, g, b\}$, and $i \neq j \neq k$.
    If $|\zeta_i - \zeta_j| \leq Th$, and $|\zeta_j - \zeta_k| \leq Th$,
            BS accepts the three aggregated results and computes the final result $M = M_i + M_j + M_k$;
    else if $|\zeta_i - \zeta_j| \leq Th$, and $|\zeta_j - \zeta_k| > Th$,
            BS rejects $M_k$, and computes the final aggregated result $M = 3/2(M_i + M_j)$;
    else if $|\zeta_i - \zeta_j| > Th$, and $|\zeta_j - \zeta_k| > Th$,
            BS either decides which aggregated result is real through gathering topology information,
            or rejects all the aggregated result $M_i$, $M_j$ and $M_k$, and return NULL.
```

The scalar of $\mathcal{P}$ is the aggregated messages, the scalar of $\mathcal{Q}$ is the count of ciphertexts, and the scale of $\mathcal{H}$ is randomness for security. We can check the integrity of the aggregated results by its count, the detail of check method is described in phase 6. For each subtree, the Public key is $PK = (n, E, \mathcal{P}, \mathcal{Q}, \mathcal{H})$ and the Private key is $SK = \{(q_1q_3), (q_2q_3)\}$.

4.2. Aggregation Tree Disjoint Phase

Three subtrees are built in this scheme, which are called red aggregation tree, green aggregation tree, and blue aggregation tree, respectively, and the BS is the root of the above three subtrees. Assuming the network is dense enough, each node, except the BS, takes one of the four roles: red aggregator, green aggregator, blue aggregator, or leaf node. We partition the tree into three subtrees, the disjoint tree is as shown in Figure 1, where the black colored nodes represent red aggregators, grey colored nodes represent green aggregators, and white colored nodes represent blue aggregators.

Figure 1. Disjoint tree construction.



Step 1. BS is appointed to be the root of the above three subtrees, which initiates a *HELLO* message requesting sensors to organize into one of the three aggregation trees. In that message, it contains its own ID and its level information $L_r = L_g = L_b = 0$.

Step 2. Each sensor receiving the message should make the decision on its role, assign its own level to be $L_i + 1(i = r, g, b)$, and select the sender node as its parent. A node becomes a red aggregator with probability p_r, a green aggregator with probability p_g, and a blue aggregator with probability p_b, respectively. The probability will be subject to the conditions: $0 < p_r = p_g = p_b < 1$, and $p_r + p_g + p_b = 1$.

Step 3. Each node in one aggregation tree rebroadcasts the colored message corresponding tree, which contains its own ID and level. If any node has already been in the tree when receives the message, it will reject the message; otherwise, the node also assigns its level L_i to be $L_i + 1$. Three aggregation trees are constructed till all nodes have a level and a parent. To balance the red, green and blue aggregators in a given neighborhood, a node should wait enough time to receive *HELLO* messages from red, green and blue aggregators as much as possible before the decision on its color is made. Then, p_r, p_g and p_b can be computed by each node as follows:

$$p_r = \frac{1}{2}\cdot\frac{N_g + N_b}{N_r + N_g + N_b},\ p_g = \frac{1}{2}\cdot\frac{N_r + N_b}{N_r + N_g + N_b},\ p_b = \frac{1}{2}\cdot\frac{N_r + N_g}{N_r + N_g + N_b} \tag{3}$$

where N_i is the number of *HELLO* messages that one sensor receives from the i aggregators ($i = r, g, b$). It should be noted that only a very few nodes do not participate in data aggregation when the network is dense enough.

Step 4. During the process of aggregation, red aggregators are not allowed to forward the data for green and blue aggregators, and *vice versa*. Then, the separation of data aggregation can be achieved along the disjoint trees. Finally, the BS will receive three aggregated results M_r, M_g and M_b respectively.

Note that an adversary may compromise the data integrity during this phase by sending two *HELLO* messages with different colors. This can be prevented by guaranteeing that a node in one tree cannot be in another two trees. However, such attack can be detected easily by its neighbors because of the shared-medium nature of wireless links. Therefore, the adversary can be excluded from the three aggregation trees.

4.3. Encryption Phase

We set $T_M < q_1$. The message space of a sensor node M should subject to $M_i \in \{0, 1, \dots, T_M\}$, where $i = r$, g and b. Each sensor picks a random $R_i \in \{0, 1, \dots, n - 1\}$, and encrypt the message M_i using public key PK, then it generates the ciphertext $C_i = M_i * \mathcal{P} + \mathcal{Q} + R_i * \mathcal{H}$, where + is the addition of elliptic curve points and $*$ is the scalar multiplication of elliptic curve.

4.4. Aggregation Phase

Let $\sum M_{ij}$ denote the aggregated message of tree T_i, ζ_i denote the number of aggregated ciphertexts of tree T_i, and $\sum R_{ij}$ denote the aggregated randomness in tree T_i, consequently, k ciphertexts for $j = 1$ to k are aggregated into a ciphertext of C_{ia} as follows:

$$C_{ia} = (\textstyle\sum_{j=1}^{k} M_{ij}) * \mathcal{P} + \zeta_i * \mathcal{Q} + (\sum_{j=1}^{k} R_{ij}) * \mathcal{H} \quad (4)$$

4.5. Decryption Phase

During the decryption phase, the BS can separately decrypt the aggregated result M_i and its count ζ_i from the aggregated ciphertext in tree T_i respectively as follows:

$$M_i = \sum M_{ij} = \log_{\tilde{\mathcal{P}}}(q_2 q_3 * C_i), \text{ where } \tilde{\mathcal{P}} = q_2 q_3 * \mathcal{P} \quad (5)$$

$$\zeta_i = \log_{\tilde{\mathcal{Q}}}(q_1 q_3 * C_i), \text{ where } \tilde{\mathcal{Q}} = q_1 q_3 * \mathcal{Q} \quad (6)$$

4.6. Data Integrity Check Phase

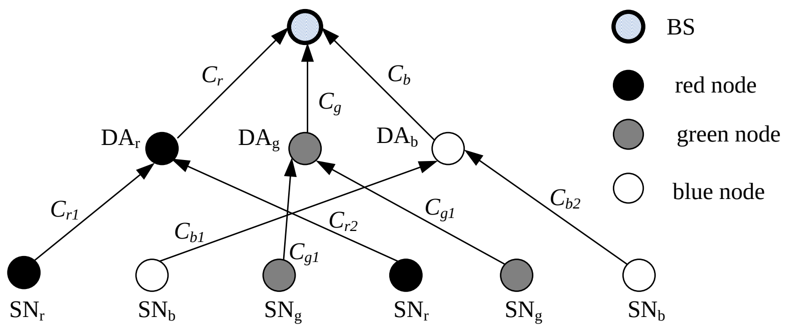
When the BS receives the three aggregated results from the red, green and blue subtrees, it should decrypt them and extract the count ζ_i, respectively. If a compromised aggregator tampers with the aggregated result $\mathrm{M_i}$, the count value ζ_i must be changed simultaneously because the aggregators do not know the base point $\mathcal{P}$ and $\mathcal{Q}$. Therefore, the BS will compare ζ_i with each other, and it indicates that the messages have not been tampered with en route only if they are almost the same. We set Th as difference threshold parameter, $i, j, k \in \{r, g, b\}$, and $i \neq j \neq k$.

If $|\zeta_i - \zeta_j| \leq Th$, and $|\zeta_j - \zeta_k| \leq Th$, it shows each result has not been tampered, then the BS accepts the three aggregated results and computes the final result $M = M_i + M_j + M_k$; if $|\zeta_i - \zeta_j| \leq Th$, and $|\zeta_j - \zeta_k| > Th$, it shows M_k has been tampered, then the BS rejects M_k, and computes the approximate aggregated result = $3/2(M_i + M_j)$; if $|\zeta_i - \zeta_j| > Th$, and $|\zeta_j - \zeta_k| > Th$, it shows the three aggregated results maybe have been tampered totally, then the BS either rejects all the aggregated results M_i, M_j and M_k, or decides which aggregated result is real by gathering topology information.

4.7. A Case Study

We present a case study to show how SEDA-ECC works. For simplicity, we assume that the network only consists of six leaf nodes and three aggregators besides BS, and the three subtrees have the same public key $PK = (n, E, \mathcal{P}, \mathcal{Q}, \mathcal{H})$. As shown in Figure 2, each subtree has two sensor nodes and one aggregator. Three aggregators, DA_r, DA_g and DA_b are deployed to gather messages from their child nodes respectively. For simplicity, the order of $\mathcal{P}$, $\mathcal{Q}$ and $\mathcal{H}$ are set to small numbers. Supposing the order of $\mathcal{P}$ and value of q_1 is 13, the order of $\mathcal{Q}$ and value of q_2 is 17, and the order of $\mathcal{H}$ and value of q_3 is 19, then the order of $n = q_1q_2q_3$ is 4,199. Sensors in three subtrees encrypt and send their data as follows, where the scalars of $\mathcal{H}$ are randomly generated by sensors.

Figure 2. A case study of SEDA-ECC.



SN_{r1} generates message $\text{M}_{r1} = 2$, and encrypts message as $\text{C}_{r1} = 2\mathcal{P} + \mathcal{Q} + 34\mathcal{H}$;
SN_{r2} generates message $\text{M}_{r2} = 5$, and encrypts message as $\text{C}_{r2} = 5\mathcal{P} + \mathcal{Q} + 13\mathcal{H}$;
SN_{g1} generates message $\text{M}_{g1} = 6$, and encrypts message as $\text{C}_{g1} = 6\mathcal{P} + \mathcal{Q} + 59\mathcal{H}$;
SN_{g2} generates message $\text{M}_{g2} = 3$, and encrypts message as $\text{C}_{g2} = 3\mathcal{P} + \mathcal{Q} + 22\mathcal{H}$;
SN_{b1} generates message $\text{M}_{b1} = 4$, and encrypts message as $\text{C}_{b1} = 4\mathcal{P} + \mathcal{Q} + 62\mathcal{H}$;
SN_{b2} generates message $\text{M}_{b2} = 15$, and encrypts message as $\text{C}_{b2} = 5\mathcal{P} + \mathcal{Q} + 39\mathcal{H}$.

The encrypted messages are sent to data aggregators. Data aggregator DA_r aggregates C_{r1} and C_{r2} as $\text{C}_r = 7\mathcal{P} + 2\mathcal{Q} + 47\mathcal{H}$. Similarly, data aggregator DA_g aggregates C_{g1} and C_{g2} as $\text{C}_g = 9\mathcal{P} + 2\mathcal{Q} + 81\mathcal{H}$, data aggregator DA_b aggregates C_{b1} and C_{b2} as $\text{C}_b = 9\mathcal{P} + 2\mathcal{Q} + 101\mathcal{H}$. Because the order of $\mathcal{H}$ is 19, $19\mathcal{H} = \infty$, where ∞ is the additive unit element in ECC. Therefore, we can get $\text{C}_r = 7\mathcal{P} + 2\mathcal{Q} + 9\mathcal{H}$, $\text{C}_g = 9\mathcal{P} + 2\mathcal{Q} + 5\mathcal{H}$, and $\text{C}_b = 9\mathcal{P} + 2\mathcal{Q} + 6\mathcal{H}$.

The aggregated result of red subtree $M_r = M_{r1} + M_{r2} = 7$ can be obtained by decrypting C_r as follows:

(1) Compute $q_2 q_3 *$ $C_r = 323*(7\mathcal{P} + 2\mathcal{Q} + 9\mathcal{H}) = 2{,}261\mathcal{P} = 12\mathcal{P}$, where $13\mathcal{P} = 17\mathcal{Q} = 19\mathcal{H} = \infty$.

(2) $M_r = \log_{\tilde{\mathcal{P}}}(q_2 q_3 * C_r) = \log_{\tilde{\mathcal{P}}} 12\mathcal{P}$, where $\tilde{\mathcal{P}} = q_2 q_3 * \mathcal{P} = 323\mathcal{P} = 11\mathcal{P}$. Then $M_r * \tilde{\mathcal{P}} = M_r * 11\mathcal{P} = 12\mathcal{P}$ can be obtained because $M_r = \log_{\tilde{\mathcal{P}}} 12\mathcal{P}$.

(3) Finally, the aggregated result of red subtree $M_r = 7$ can be obtained by the BS according to Pollard's λ method.

Similarly, the BS can also extract the aggregated count result ζ by computing the discrete logarithm of $q_1 q_3 * C_r$ to the base point $\tilde{\mathcal{Q}} = q_1 q_3 * \mathcal{Q}$. Therefore, BS can identify the forged result by comparing the aggregated count value. If the difference among three subtrees aggregated results is within the range of threshold *Th*, then BS validates the integrity of the aggregated result.

5. Theoretical Analysis

In this section, we analyze the coverage of aggregation trees first because it has great effect on our scheme's availability, then analyze the security of SEDA-ECC and compare it with five well-known secure data aggregation schemes: CDA [16,17], Castelluccia *et al.*'s scheme [22], BGN scheme [11], EC-OU scheme [18], and TinyPEDS scheme [8].

5.1. Coverage of Aggregation Trees

In SEDA-ECC scheme, a sensor reports its data to BS by aggregation only when it can reach red, green and blue aggregation trees within one hop. If a node cannot reach the three aggregation trees, it is disconnected from the BS for aggregation. We define $\Phi(G)$ as the probability that all the sensors are covered by all the three aggregation trees. It means that many sensors cannot contribute their data to the aggregation result if $\Phi(G)$ is small. Therefore, the coverage of aggregation trees impacts the accuracy of aggregation results. The aggregation accuracy is one of the most important performance metrics, because it can affect the decision of BS, so we should first analyze the coverage of aggregation trees to verify our scheme's availability.

Consider a random network G (n, l), where n is the number of sensors, and l is the transmission range of a sensor. We randomly assign red, green or blue to sensors in the networks, and let S denote the number of sensors which are isolated from red, green or blue sensors, then:

$$\Phi(G) = P(S = 0) \tag{7}$$

We define S_i as the variable of whether sensor i has red, green and blue neighbors within one hop distance, then

$$S_i = \begin{cases} 0, & i \text{ has red, green and blue neighbors} \\ 1, & \text{otherwise} \end{cases} \tag{8}$$

$\{S_i\}$ can be approximated as identical independent distributions for a random network whose size is large enough, therefore, $S = \sum_{i=1}^{n} S_i$ can be denoted as the total number of sensors which are isolated by red, green or blue aggregation tree. Let d_i denote the number of neighbors of sensor i, then the probability that i is isolated by the red aggregation tree is labeled as $p_{bg}^{d_i}$. Similarly, i is isolated by the green (blue) aggregation tree with the probability $p_{rb}^{d_i}(p_{rg}^{d_i})$. Let p_i be the probability that note i is isolated by red sensors, green sensors or blue sensors, then:

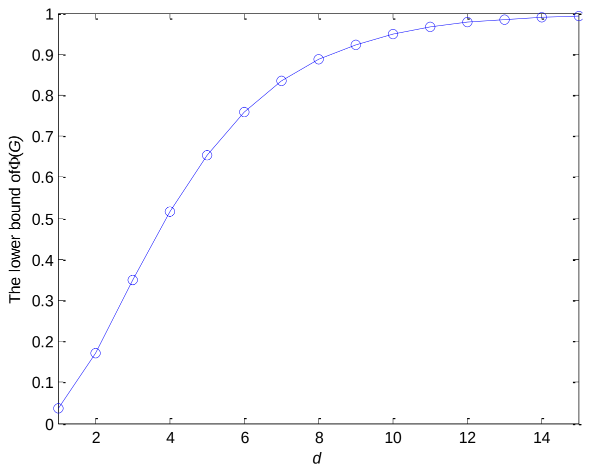
$$p_i = P(S_i = 1) = 1 - (1 - p_{bg}^{d_i})(1 - p_{rb}^{d_i})(1 - p_{rg}^{d_i}) \tag{9}$$

Since $S = \sum_{i=1}^{n} S_i$, we can get a lower bound of Φ(*G*) by applying Markov Inequality $P(S \geq 1) \leq E[S] = \sum_{i=1}^{n} p_i$. That is:

$$\Phi(G) \geq \frac{n - \sum_{i=1}^{n} p_i}{n} \tag{10}$$

When the network is dense enough, *i.e.*, d_i is large, a small p_i can be obtained. For example, assuming $p_r = p_g = p_b = 1/3$, we can obtain the lower bound of Φ(*G*) which varies with the variation of *d* under the condition of the *d*-regular network according to Equation (9). It can be observed from Figure 3 that Φ(*G*) ≥ 0.95 for *d* = 10, therefore, the coverage of aggregation trees is perfect for dense networks from Equation (10).

Figure 3. The lower bound of Φ(*G*) which varies with the variation of *d*.



5.2. Ciphertext Analysis

This is the most basic attack in WSNs. SEDA-ECC is robust to ciphertext analysis attack, because the elliptic curve cryptography-based encryption depends on the factorization of large integers. Other schemes are also robust to ciphertext analysis attacks.

5.3. Chosen Plaintext Attacks

SEDA-ECC is robust to chosen plaintext attacks, because its encryption relies on random numbers, and the ciphertext is probabilistic. Other schemes based on ECC can defend against this attack too. Wagner's cryptanalysis [23] has indicated that CDA might suffer from chosen plaintext attacks because of improper security parameters. However, the cost of proper parameter would render CDA infeasible to WSNs. Castelluccia *et al.*'s scheme is also robust to this attack, because its security is based on the indistinguishability property of a pseudorandom function, and the previous encryption keys cannot be used to deduce the present or subsequent encryption key.

5.4. Malleability

In the analysis of this attack, we give the example that the adversary wants to increase the measured data by 50. Since Castelluccia *et al.*'s scheme is based on modular addition, adversaries can add the value of plaintext trivially through adding a certain value to the corresponding ciphertext directly, so it suffers from this attack. For example, a ciphertext $(m + K_n) \bmod M$ can be easily altered by $((m + 50) + K_n) \bmod M = (m + K_n) + 50 \bmod M$. Other schemes can defend against this attack because they are based on either modular multiplication or ECC.

5.5. Unauthorized Aggregation

For asymmetric scheme, SEDA-ECC, BGN, EC-OU and TinyPEDS are based on ECC. If an aggregator needs to perform aggregation, it has to know curve information. Since the public key is preinstalled in sensors generally, adversary cannot perform unauthorized aggregation and falsify the aggregated count value of subtrees without compromising the sensors or aggregators. CDA and Castelluccia *et al.*'s scheme might suffer from this attack, because they require only modular addition, and unauthorized aggregation can be performed without any additional information.

5.6. Node Compromise Attacks

For asymmetric schemes, SEDA-ECC, BGN, EC-OU and TinyPEDS do not suffer from unauthorized decryption under compromised sensor node conditions, because an adversary cannot obtain the private key through a compromised sensor. However, except for SEDA-ECC, they cannot defend against unauthorized aggregation in a compromised aggregator situation. The compromised aggregator might arbitrarily increase the aggregated result by aggregating the same ciphertext repeatedly or decrease it by selective aggregation. After the aggregation process, the forged value is difficult to detect or remove by the BS. SEDA-ECC can prevent this attack targeting data integrity by constructing disjoint aggregation subtrees. It is impossible for attackers to alter the aggregated result M without changing the count value ζ because the aggregators do not know the base points $\mathcal{P}$ and $\mathcal{Q}$. If the aggregated result of one tree is different from the others, the BS will reject it and compute the final result from the others. Therefore, an attacker can successfully forge the aggregated result if and only if the forged aggregated results of two trees are the same. The probability of success is extremely small, because the security depends on the factorization of large integers.

We use the case study of SEDA-ECC in Section 4.7 to validate its ability of defending against this attack. Supposing the aggregation ciphertexts excluding C_r, C_g, and C_b are $C'_r = M'_r\mathcal{P} + 193\mathcal{Q} + R_r\mathcal{H}$, $C'_g = M'_g\mathcal{P} + 190\mathcal{Q} + R_g\mathcal{H}$, and $C'_b = M'_b\mathcal{P} + 191\mathcal{Q} + R_b\mathcal{H}$. If the red aggregator DA_r is compromised, it can arbitrarily increase the aggregated result by aggregating the same ciphertext repeatedly. Supposing the compromised aggregator DA_r intend to increase C_r by aggregating C_{r1} 20 times, then $C_r = 20C_{r1} + C_{r2} = 45\mathcal{P} + 21\mathcal{Q} + 693\mathcal{H}$. Therefore, we can get the aggregation ciphertext results $\mathbb{C}_r = C'_r + C_r = (M'_r + 45) + 214\mathcal{Q} + (R_r + 693)\mathcal{H}$, $\mathbb{C}_g = C'_g + C_g = (M'_g + 9)\mathcal{P} + 192\mathcal{Q} + (R_g + 5)\mathcal{H}$, and $\mathbb{C}_b = C'_b + C_b = (M'_b + 9)\mathcal{P} + 193\mathcal{Q} + (R_b + 6)\mathcal{H}$, respectively. When the aggregated count results $(\mathrm{r}_r, \zeta_g, \zeta_b)$ are extracted by computing the discrete logarithm of $q_1q_3*(\mathbb{C}_r,\mathbb{C}_g,\mathbb{C}_b)$ to the base point $\tilde{\mathcal{Q}} = q_1q_3*\mathcal{Q}$, the forged result $\mathbb{C}_r$ can be easily identified and rejected by BS because the differences between ζ_r and the other two are out of the threshold value *Th*, that is $|\zeta_r - \zeta_g| = 22 > Th$, and $|\zeta_r - \zeta_b| = 21 > Th$.

For symmetric schemes, the inherent drawback of CDA and Castelluccia *et al.*'s schemes is that the encryption key is identical with the decryption key. Therefore, an adversary can decrypt the ciphertext once the sensor is compromised. In addition, because the CDA's key is shared by all sensors and BS, if any sensor is compromised, the whole system security is broken. Castelluccia *et al.*'s scheme suffers from a minor impact due to the fact its distinct key is shared by BS.

Table 1 shows the security analysis comparisons for all schemes. It clearly shows that symmetric schemes are less secure than asymmetric ones, although they are more efficient in terms of communication and computation costs. Compared with other asymmetric schemes, SEDA-ECC is superior in defending against compromised node attacks because it can protect data integrity by constructing disjoint aggregation trees when the aggregators are compromised.

Table 1. Security comparisons.

Requirement	Ciphertext Analysis	Chosen Plaintext Attacks	Malleability	Unauthorized Aggregation	Node Compromise Attacks
CDA	√	×	√	×	×
Castelluccia *et al.*'s scheme	√	√	×	×	×
SEDA-ECC	√	√	√	√	√
BGN	√	√	√	√	×
EC-OU	√	√	√	√	×
TinyPEDS	√	√	√	√	×

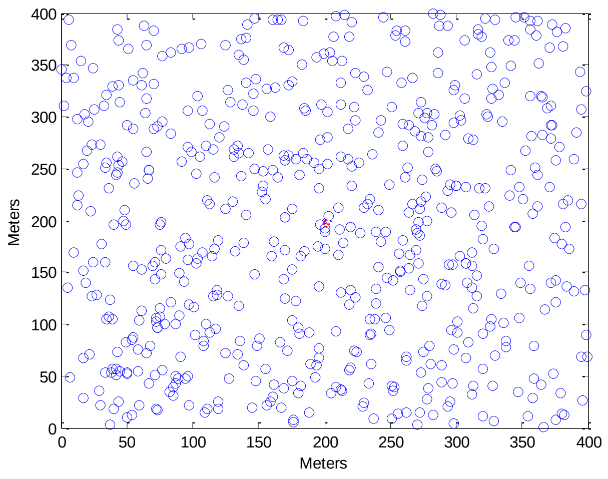
6. Performance Evaluation and Comparison

Generally, symmetric key-based homomorphic schemes are more efficient than asymmetric ones, however, the security of symmetric schemes is weaker than that of asymmetric ones. For the sake of fairness, the performance of SEDA-ECC is only compared with other three asymmetric key-based homomorphic encryption schemes. In this section, we first discuss the threshold value *Th*, then evaluate the computation overhead, communication cost, and the accuracy of SEDA-ECC, BGN, EC-OU and TinyPEDS. We conduct simulations using TinyOS 2.0 simulator (TOSSIM). The parameters are shown in Table 2, and the topology of nodes is depicted in Figure 4, where the transmission range of a sensor is 50 m, and the BS coordinate is (200,200).

Table 2. Simulation parameters.

Radio Parameters		Topology Parameters	
Noise Floor	**White Gaussian Noise**	**Terrain Dimensions**	**Number of Nodes**
−105 dB	4 dB	400 m × 400 m	600

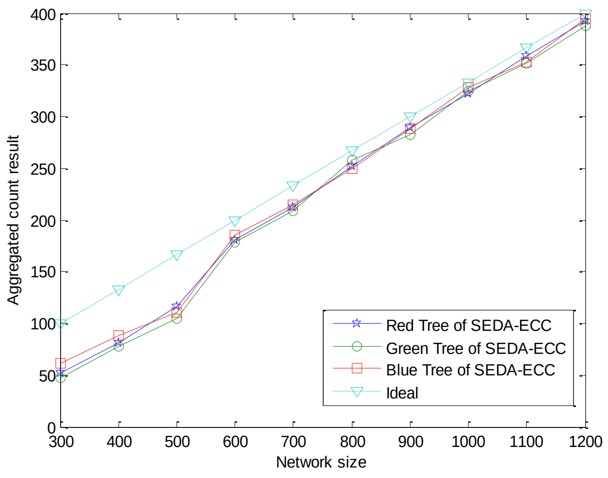
Figure 4. Nodes distribution.



6.1. Th Parameter Setting

In general, the more sensors that participate in the data aggregation, the larger the probability of constructing disjoint aggregation trees which have the same number of sensors. In addition, the aggregated count results ζ from three aggregation trees may not agree with each other exactly due to collisions and congestions in wireless channels. Therefore, an adjustable threshold value *Th* and the lowest bound of network size are introduced to accomodate these factors. Since whether the BS accepts the result depends on the threshold value *Th*, hence *Th* is an important parameter. In order to get *Th*, we did extensive simulations, where the number of nodes (network size) was varied from 300 to 1,200 in a 400 m × 400 m area.

Figure 5. Difference value among aggregated count results from three subtrees without attack.

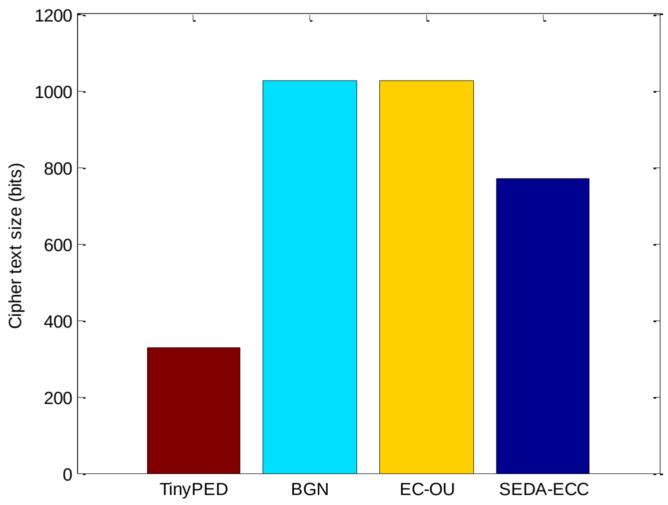


The difference value among aggregated count results from three aggregation subtrees is simulated 40 times, and the average value is depicted in Figure 5, where the "ideal" curve shows the aggregated result in an ideal situation. According to the simulation result, we notice that the differences, which are between 2 and 9, are very small. Hence, the threshold can be set as a small value, e.g., *Th* = 10. We can adjust *Th* if the network conditions are changed. Note that the average count result is only half of the ideal number and the difference extends to 9 when the network size is nearly 300. In addition, the smaller network size is, the larger differences became. As we analyzed in Section 5.1, it is because the coverage is bad enough in a sparse network to deteriorate the aggregation accuracy. Therefore, we set the lowest bound of network size as 300 in a 400 m × 400 m area to make our scheme available.

6.2. Communication Overhead

The number of exchanged messages in each scheme is the same. Though there are three subtrees need to be built in SEDA-ECC, similar to the other schemes, each node needs to send two messages for data aggregation: one *HELLO* message to form the aggregation tree, and the other message for data aggregation. Therefore, the communication overhead mainly depends on the ciphertext size of each scheme on the condition that the number of message sending to the BS is the same. Supposing the order of elliptic curve is N, SEDA-ECC's security relies on the hardness of factoring the order N. N is a product of several different large prime numbers, e.g., $N = q_1q_2 \cdots q_k$, where k is the number of prime numbers. If the length of prime number is 256-bit, there is no efficient approach to factor the product N [7]. Therefore, in SEDA-ECC, we generate $N = q_1q_2q_3$, where the prime numbers q_i are all 256-bit. Since the size of the ciphertext is almost the same as $|N| + 1$, the SEDA-ECC's ciphertext size is $3 \times |q| + 1(|q| = 256\text{-bit})$. EC-OU's ciphertext size is $3 \times |q| + 2(|q| = 341\text{-bit})$ according to [24]. BGN's ciphertext size is 1,025-bit, and TinyPEDS's ciphertext size is 328-bit according to [7]. Figure 6 shows the comparison of ciphertext sizes.

Figure 6. The comparison of ciphertext sizes.

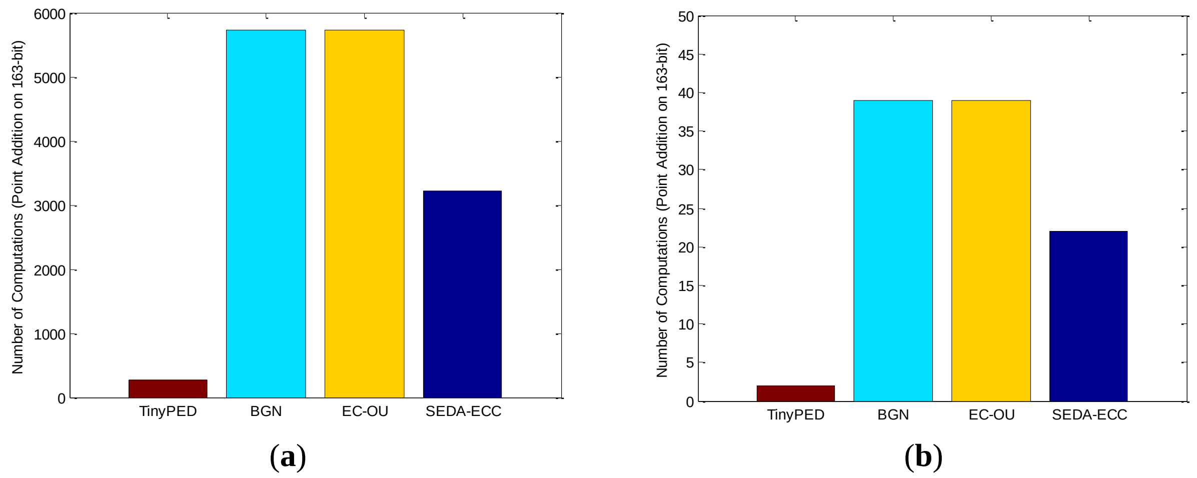


6.3. Computation Overhead

Since SEDA-ECC, BGN, EC-OU and TinyPEDS schemes are all built on elliptic curves, encryption and aggregation operation are based on point addition and point scalar multiplication. In elliptic curve arithmetic, point doubling and adding are two basic operations. Scalar multiplication can be accomplished by the half-and-add algorithm based on point doubling and adding [25]. It requires about $|r|$ doubling and $|r|/2$ additions for computing $r*Q$, amounting to around $3|r|/2$ point additions [18].

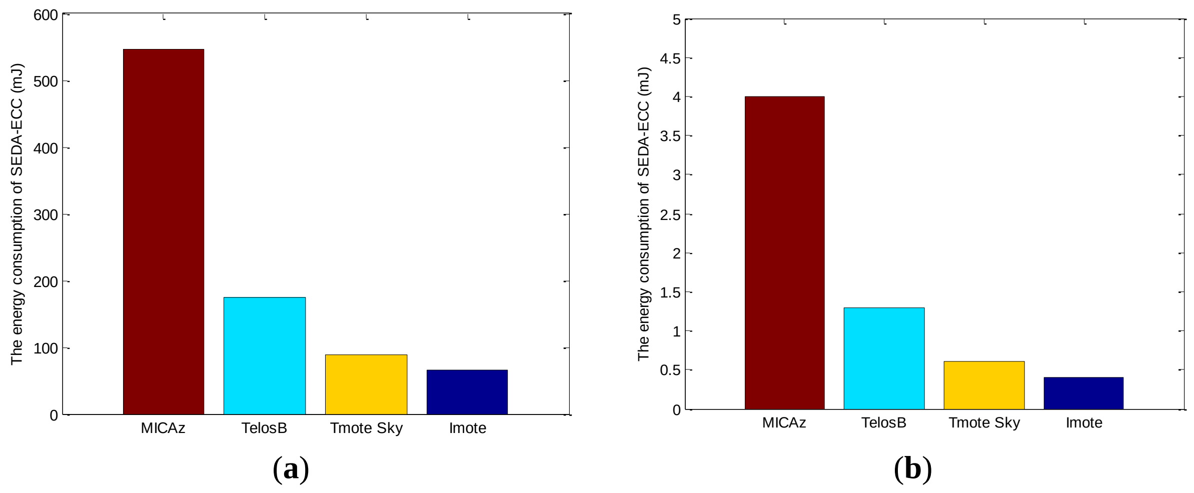
It should be noted that SEDA-ECC, BGN, EC-OU and TinyPEDS schemes are built on different mathematical foundations. We assume the finite field of elliptic curve is $\mathcal{F}_p$, and the bit length of the finite field is |p|. BGN and EC-OU schemes are chosen over $\mathcal{F}_p$ (|p| = 1,024), TinyPEDS is chosen over $\mathcal{F}_p$ (|p| = 163), SEDA-ECC is chosen over $\mathcal{F}_p$ (|p| = 768). To achieve a fair comparison, we choose the point addition on 163-bit field as the base unit. For an elliptic curve computation over a finite field $\mathcal{F}_p$, the cost of scalar multiplication can be converted to the number of computations (point addition on 163-bit) according to the scalar r and the size |p|. The comparison results are presented in Figure 7, where the length of messages is 16-bit, and the length of random nonces is 80-bit.

Figure 7. The comparison of cost; (**a**) encryption cost; and (**b**) aggregation cost.



In summary, TinyPED is the most efficient one for both communication overhead and computation cost, because its curves are chosen from the smaller field $\mathcal{F}_p$ (|p| = 163). TinyPED's security is based on the hardness of elliptic curve discrete logarithm problem, hence it can be built on a smaller field. However, BGN, EC-OU and SEDA-ECC are all based on the hardness of integer factorization problem, so their curves must be chosen from the larger field. It can also be observed from Figures 6 and 7 that SEDA-ECC outperforms BGN and EC-OU for both communication and computation performances. Furthermore, In terms of security, SEDA-ECC can defend against all attacks which are listed in Table 1, hence it is superior to the other schemes.

Figure 8. The energy consumption of SEDA-ECC in different sensor devices; (**a**) encryption consumption and (**b**) aggregation consumption.

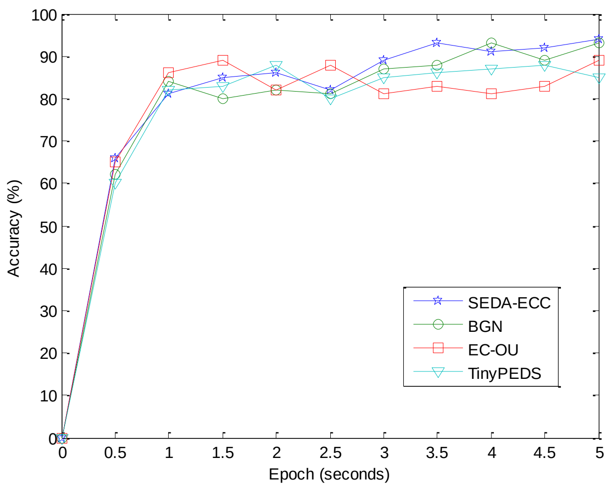


Furthermore, the energy consumption of SEDA-ECC is evaluated in different sensor devices according to TinyECC [26], which is one well-known implementation of ECC for WSNs, as shown in Figure 8. The energy consumption can be significantly reduced with more advanced devices. Therefore, the secure data aggregation schemes based on asymmetric encryption, e.g., ECC, have extensive applications with the development of the advanced sensors.

6.4. Accuracy

We define the accuracy as the ratio between the aggregated sum result by the data aggregation scheme in use and the real sum of all sensors participating in the data aggregation. It is an important issue because it could affect the decision of the BS. All the schemes should achieve 100% accurate aggregated results in an ideal situation. However, data packets may be lost or delayed due to data collisions, processing delays and noisy wireless channels. We evaluate the data accuracy of SEDA-ECC, BGN, EC-OU, and TinyPED with respect to different time intervals, as shown in Figure 9.

Figure 9. The comparison of accuracy with respect to different time intervals.



It shows that all these schemes almost perform equally in term of accuracy. We can observe that the accuracy increases as the time interval increases, because the data collisions and congestions between data aggregators are reduced, and the data packets should have enough time to be delivered.

7. Conclusions

Providing hierarchical data aggregation without losing data privacy and integrity guarantee is a challenging problem in WSNs. In this article, we propose a novel Secure-Enhanced Data Aggregation based on Elliptic Curve Cryptography (SEDA-ECC) for WSNs. SEDA-ECC divides the aggregation tree into three subtrees to reduce the importance of the high-level sensor nodes. It also generates three aggregated results by performing PH-based aggregations in the three subtrees, respectively, so that the BS could verify the subtree aggregated results by comparing the aggregated count value. Extensive analytical and simulation results indicate that SEDA-ECC can achieve the highest security level on the aggregated result comparing with other asymmetric schemes, and SEDA-ECC is efficient with respect to a reasonable energy cost.

Acknowledgments

This work is supported by the National Basic Research Program (973 Program) of China under Grant No. 2011CB302903, the National Natural Science Foundation of China under Grant Nos. 61272084 and 61300240, the National Research Project of China under Grant No. 2012A138, the Key Project of the Natural Science Research of Jiangsu University under Grant No. 11KJA520002, the Research Innovation Program for College Graduates of Jiangsu Province under Grant No. CXZZ12_0477, the Key Foundation for Outstanding Young Talent of Chuzhou University under Grant No. 2013RC002, and the Natural Science Foundation of Chuzhou University under Grant Nos. 2012kj003Z and 2011kj014B.

Author Contributions

All authors have contributed to the presented work in various degrees. Qiang Zhou and Geng Yang proposed the network model and scheme. Qiang Zhou performed theoretical analysis and wrote the manuscript. Qiang Zhou and Liwen He conducted the experimental simulations, and revised the manuscript. All authors have read and approved the final manuscript.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Madden, S.; Franklin, M.J.; Hellerstein, J.; Hong, W. TAG: A tiny aggregation service for ad-hoc sensor networks. In Proceedings of the 5th Operating Systems Design and Implementation Symposium (ACM OSDI ’02), Boston, MA, USA, 9–11 December 2002; pp. 131–146.
2. Fasolo, E.; Rossi, M.; Widmer, J.; Zorzi, M. In-network aggregation techniques for wireless sensor networks: A survey. *IEEE Wirel. Commun.* **2007**, *14*, 70–87.
3. Yang, Y.; Wang, X.; Zhu, S.; Cao, G. SDAP: A secure hop-by-hop data aggregation protocol for sensor networks. *ACM Trans. Inf. Syst. Secur. TISSEC* **2008**, *11*, 1–43.
4. Yang, G.; Li, S.; Xu, X.; Dai, H.; Yang, Z. Precision-enhanced and encryption-mixed privacy-preserving data aggregation in wireless sensor networks. *Int. J. Distrib. Sens. Netw.* **2013**, *2013*, 427275.
5. Jung, T.; Mao, X.F.; Li, X.Y.; Tang, S.J.; Gong, W.; Zhang, L. Privacy-preserving data aggregation without secure channel: Multivariate polynomial evaluation. In Proceedings of the 32th Conference on Computer Communications (IEEE INFOCOM ’13), Turin, Italy, 14–19 April 2013; pp. 2734–2742.
6. Wang, L.; Wang, L.; Pan, Y.; Zhang, Z.; Yang, Y. Discrete logarithm based additively homomorphic encryption and secure data aggregation. *Inf. Sci.* **2011**, *181*, 3308–3322.
7. Lin, Y.; Chang, S.; Sun, H. CDAMA: Concealed data aggregation scheme for multiple applications in wireless sensor networks. *IEEE Trans. Knowl. Data Eng.* **2012**, *25*, 1471–1483.
8. Girao, J.; Westhoff, D.; Mykletun, E.; Araki, T. TinyPEDS: Tiny persistent encrypted data storage in asynchronous wireless sensor networks. *Ad Hoc Netw.* **2007**, *5*, 1073–1089.

9. Ozdemir, S.; Cam, H. Integration of false data detection with data aggregation and confidential transmission in wireless sensor networks. *IEEE CM Trans. Netw. TON* **2010**, *18*, 736–749.
10. Papadopoulos, S.; Kiayias, A.; Papadias, D. Exact In-network aggregation with integrity and confidentiality. *IEEE Trans. Knowl. Data Eng.* **2012**, *24*, 1760–1773.
11. Boneh, D.; Goh, E.J.; Nissim, K. Evaluating 2-DNF formulas on ciphertexts. In Proceedings of the 2nd International Conference on Theory of Cryptography (TCC '05), Cambridge, MA, USA, 10–12 February 2005; pp. 325–341.
12. Chan, H.; Perrig, A.; Song, D. Secure hierarchical in-network aggregation in sensor networks. In Proceedings of the 13th ACM Conference on Computer and Communications Security (ACM CCS '06), Alexandria, VA, USA, 30 October–3 November 2006; pp. 278–287.
13. Frikken, K.B.; Dougherty, J.A., IV. An efficient integrity-preserving scheme for hierarchical sensor aggregation. In Proceedings of the first ACM Conference on Wireless Network Security (ACM WISEC '08), Alexandria, VA, USA, 31 March–2 April 2008; pp. 68–76.
14. Zhu, L.; Yang, Z.; Li, M.; Liu, D. An Efficient data aggregation protocol concentrated on data integrity in wireless sensor networks. *Int. J. Distrib. Sens. Netw.* **2013**, *2013*, 256852.
15. Niu, S.; Wang, C.; Yu, Z.; Cao, S. Lossy data aggregation integrity scheme in wireless sensor networks. *Comput. Electr. Eng.* **2013**, *39*, 1726–1735.
16. Westhoff, D.; Girao, J.; Acharya, M. Concealed data aggregation for reverse multicast traffic in sensor networks: Encryption, key distribution, and routing adaptation. *IEEE Trans. Mob. Comput.* **2006**, *5*, 1417–1431.
17. Girao, J.; Westhoff, D.; Schneider, M. CDA: Concealed data aggregation for reverse multicast traffic in wireless sensor networks. In Proceedings of the IEEE International Conference on Communications (IEEE ICC '05), Seoul, Korea, 16–20 May 2005; pp. 3044–3049.
18. Mykletun, E.; Girao, J.; Westhoff, D. Public key based cryptoschemes for data concealment in wireless sensor networks. In Proceedings of the IEEE International Conference on Communications (IEEE ICC '06), Istanbul, Turkey, 11–15 June 2006; pp. 2288–2295.
19. Cheon, J.H.; Kim, W.H.; Nam, H.S. Known-plaintext cryptanalysis of the Domingo-Ferrer algebraic privacy homomorphism scheme. *Inf. Process. Lett.* **2006**, *97*, 118–123.
20. Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT '99), Prague, Czech Republic, 2–6 May 1999; pp. 223–238.
21. Okamoto, T.; Uchiyama, S. A new public-key cryptosystem as secure as factoring. In Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT '98), Helsinki, Finland, 31 May–4 June 1998; pp. 308–318.
22. Castelluccia, C.; Chan, A.; Mykletun, E.; Tsudik, G. Efficient and provably secure aggregation of encrypted data in wireless sensor networks. *ACM Trans. Sens. Netw. TOSN* **2009**, *5*, 1–20.
23. Wagner, D. Cryptanalysis of an algebraic privacy homomorphism. In Proceedings of the 6th International Conference on Information Security, Bristol, UK, 1–3 October 2003; pp. 234–239.
24. Peter, S.; Westhoff, D.; Castelluccia, C. A survey on the encryption of convergecast traffic with in-network processing. *IEEE Trans. Dependable Secur. Comput.* **2010**, *7*, 20–34.
25. Cohen, H.; Frey, G.; Avanzi, R. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*; CRC Press: Boca Raton, FL, USA, 2010; pp. 157–215.

26. Liu, A.; Ning, P. TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks. In Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN'08), Missouri, USA, 22–24 April 2008; pp. 245–256.