Computing Aggregation Function Minimum/Maximum using Homomorphic Encryption Schemes in Wireless Sensor Networks (WSNs)

Levent Ertaul
Department of Mathematics & Computer Science
California State University, EastBay
Hayward, CA, USA.
levent.ertaul@csueastbay.edu

Vaidehi
Department of Mathematics & Computer Science
California State University, EastBay
Hayward, CA, USA.
vaidehikedlaya@gmail.com

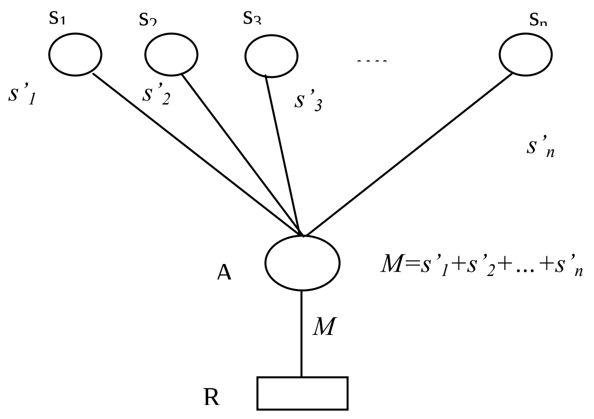
Abstract- *Data aggregation in wireless sensor networks (WSN) helps eliminate information redundancy and increase the lifetime of the network. When homomorphic encryption is used for data aggregation, end-to-end encryption is achieved and aggregation function like average or minimum/maximum can be computed on the encrypted data. Aggregation functions like minimum/maximum rely on comparison operation. But, it has been shown that any homomorphic encryption is insecure against ciphertext only attacks if they support comparison operation. The order preserving encryption scheme (OPES) has been suggested for WSNs, for secure comparison of encrypted data at the aggregator node in WSNs. But, the computational cost at the sensor nodes in WSNs by using OPES is huge. This paper provides an alternative for OPES when used to calculate aggregation function minimum/maximum. In this paper we briefly describe some homomorphic encryption schemes and show how the sensed data is encrypted by using these homomorphic encryption schemes. we show how aggregation function minimum/maximum can be computed at the aggregator node in WSNs by performing addition operation and not comparison operation on the data encrypted with homomorphic encryption schemes. We also show how our scheme helps eliminate the encryption cost at the sensor node in WSNs.*

Index Terms—Wireless sensor networks, data encryption, data aggregation, homomorphic encryption schemes.

1 Introduction

A WSN consists of less expensive and low power sensor nodes that are capable of computation, storage and communication. These sensor nodes have low computation power and storage space. The purpose of deploying a sensor node is to monitor an area of interest with respect to some physical quantity. Information gathered by the sensor nodes is reported to the base station [1].

The sensor nodes have low computation power and low storage capacity. Each sensor node senses their environment and transmits the data to the central point. The data gathered by the sensor nodes in most scenarios like environmental data (eg, temperature) will eventually be computed to find the minimum, maximum or average. These computations could be carried out at a central point or by the network itself. The latter has the advantage of reducing the amount of data transmitted over wireless connections. Since the energy consumption increases linearly with the amount of transmitted data, an aggregation approach helps increase the overall lifetime of WSN [1, 2, 3, 4, 5].



s_1 to s_n are the sensor nodes, A is the aggregator node and R is the sink node. s'_n are the value sensed by the sensors.

Figure 1: Representation of Sensor node, Aggregator node and Sink node in WSN

The authors in [2] have logically separated WSN into sensor nodes, forwarding nodes, aggregator nodes and sink node. Let us look into the Figure 1 introduced in [2, 3, 4] which explains the flow of data from sensor nodes to sink node. Sensor nodes sense the environment and send the data to the aggregator node. Aggregator node aggregates the data received from the senor nodes. The aggregation function may be calculation of the average, minimum/maximum or movement detection. The forwarding node just forwards the data. Sink node is assumed to be more powerful than the sensor nodes and the aggregator node. Aggregator node and the forwarding nodes belong to the backbone, whereas the sensor nodes persist in sleep mode until the sink node initiates a process that requires a subset of them to contribute.

The concept of concealed data aggregation (CDA) is introduced in [3, 4], where the authors address the security and energy requirements for WSNs. In CDA the authors use Domingo-Ferrer's Privacy Homomorphism [6], which provides end-to-end encryption between the sensor and the sink node. The aggregators carry out the aggregation function on the encrypted data. Privacy Homomorphism is a homomorphic encryption scheme, which allows operation to be performed on the encrypted data (ciphertext) as if the operation is performed on the plaintext. Homomorphic encryption schemes may have the property of additive or multiplicative homomorphism. In *additive homomorphism*, decrypting the sum of two ciphertext is same as addition of two plaintext (x ,y) represented as $x+y=D_k(E_k(x)+E_k(y))$. In *multiplicative homomorphism*, decrypting the product of two ciphertext is same as multiplication of two plaintext. Multiplicative homomorphism is mathematically represented as
$x*y=D_k(E_k(x)*E_k(y))$. The advantage of using homomorphic encryption is that the intermediate aggregator node need not decrypt and then encrypt to perform the aggregation operation. In paper [3, 4], the authors performed aggregation function average and movement detection using Domingo-Ferrer's privacy homomorphism [6]. The aggregation function minimum/maximum is a comparison-based function. In [7], Rivest has shown that a Privacy Homomorphism is insecure against ciphertext only attacks if it supports comparison operations. In [5] the authors apply OPES [8] for CDA, to perform secure ciphertext comparison required by the aggregation function minimum/maximum. In this paper we show that the aggregation operation minimum/maximum can be performed by computing addition operation and not comparison operation on the encrypted data at the aggregator node. This paper provides an alternative for OPES scheme when used to calculate the aggregation function minimum/maximum.

The paper is organized as follows. In section 2, we briefly describe the overview of homomorphic encryption schemes. In section 3, we briefly describe the OPES scheme used for secure ciphertext comparison. In section 4, we propose a new scheme to find the minimum/maximum at the aggregator node. Finally, in section 5 conclusions are given.

2 Encryption Schemes exhibiting the property of homomorphism

In this section we give an overview of four different encryption schemes which exhibit the property of homomorphism.

2.1 Encryption functions using Mod Operations

In this section, we focus on encryption scheme using *mod* operations, which is *cryptosystem using mod operation* exhibiting the property of homomorphism.

The ***cryptosystem using mod operation*** is introduced in [9]. This cryptosystem uses large number m, where $m= p* q$. Here p and q are large prime numbers, which are kept secret. The set of original plaintext messages is in $Z_p =\{ x|x <= p \}$, $Z_m = \{ x|x <m \}$ has the set of ciphertext messages and $Q_p = \{ a|a \in Z_p \}$ has the set of encryption clues.

The *encryption algorithm* is performed by choosing a plaintext 'x' $\in Z_p$ and a random number 'a' in Q_p such that $x = a \bmod p$. Here p is kept secret. The ciphertext y is calculated as $y = E_p (x) = a \bmod m$.

In *decryption algorithm* the plaintext x is recovered as $x= D_p(y) = y \bmod p$, where p is the secret key.

This cryptosystem has the property of additive, multiplicative and mixed multiplicative homomorphism. The proposed protocol, though exhibits the property of homomorphism is not very secure against known plaintext attacks, but secure against known ciphertext attacks [9].

2.2 Privacy Homomorphic Encryption Schemes

In this section we look into *Domingo-Ferrer's* three different *privacy homomorphism*.

Domingo-Ferrer's New Privacy homomorphism is introduced in [10] which is a homomorphic encryption scheme not vulnerable to known ciphertext attacks.

Let us look into the protocol in detail. In this protocol n and m are the public parameters. Here $m= p * q$, where p and q are large prime numbers. To increase security, m can be kept secret. The number 'n' represents the split of the plaintext. The secret keys are p, q, x_p, x_q. Here, $x_p \in Z_p$ and $x_q \in Z_q$.

Encryption operation is performed by selecting the plaintext $a \in Z_m$. We then split a into secret numbers $a_1, a_2 ... a_n$, such that $a = (a_1 + a_2 ... +a_i+...a_n) \bmod m$ and $a_i \in Z_m$.

$E_k (a) = (a_1 x_p \bmod p, a_1 x_q \bmod q), (a_2 x^2_p \bmod p, a_2 x^2_q \bmod q)... (a_n x^n_p \bmod p, a_n x^n_q \bmod q)$

Decryption operation is performed by computing scalar product of the i^{th} pair *[mod p, mod q]* by $[x^{-i}_p \; mod \; p, x^{-i}_q \; mod \; q]$ to get $[a_i \; mod \; p, a_i \; mod \; q]$. The pairs are then added up to get *[a mod p, a mod q]*. Finally, Chinese remainder theorem (CRT) [11] is performed to get *a mod m.*

The privacy homomorphism has the property of additive and multiplicative homomorphism. This homomorphism scheme though secure against know ciphertext attacks is not very secure against known plaintext attacks [12].

Domingo-Ferrer's Privacy Homomorphism allowing field operation on encrypted data is introduced in [13]. In this encryption scheme p and p' are large secret primes and let $q = pp'$ is public. Qp is defined as $Qp = \{a/b : a,b \in Zp\}$

Encryption operation is performed by selecting a value $x \in Zp$, a random fraction a/b in Qp, such that $x = ab^{-1} \; mod \; p$. The ciphertext is computed as $y = Ep(x) = ab^{-1} \; mod \; q$.

Decryption operation is performed by picking any fraction $A/B \in Qp$ such that $y = AB^{-1} \; mod \; q$. The key p is used to recover the plaintext x as $x = Dp(y) = AB^{-1} \; mod \; p$.

This privacy homomorphism has the property of additive, multiplicative and mixed multiplicative homomorphism. The privacy homomorphism is secure against chosen ciphertext attacks but not very secure against known-plaintext attacks [13].

Domingo-Ferrer's Additive and Multiplicative Privacy homomorphism is introduced in [6]. In this protocol the public parameters are $d>2$ and m. m should have many small divisors and there should be many integers less than m that can be inverted modulo m. The secret parameters are $r \in Z_m$ and m' such that $r^{-1} mod \; m$ exists and a small divisor $m > 1$ of m such that $s := log_{m'}m$ is an integer.

Encryption operation is performed by randomly splitting $a \in \mathbf{Z}_{m'}$ into secret $a_1,\cdots ,a_d$ such that $a =(a_1 + a_2 \ldots +a_i+\ldots a_n) \; mod \; m'$ and $a_i \in \mathbf{Z}_m$. Compute

$E_k(a) = (a_1 r \; mod \; m, a_2 r^2 \; mod \; m, \ldots , a_d r^d \; mod \; m)$

Decryption operation is performed by computing the scalar product of the *j-th* coordinate by $r^{-j} mod \; m$ to retrieve $a_j \; mod \; m$. The plaintext a is a obtained by computing, $(a_1+..+a_j +\ldots+ a_d) mod \; m'$.

This privacy homomorphism has the additive, subtractive, multiplicative and division homomorphism. The privacy homomorphism is secure against chosen ciphertext attacks but not secure against chosen plaintext attacks as shown by Wagner [14].

In the next section we look into OPES scheme used to perform secure comparison over encrypted data.

3 Adapting OPES scheme for Encrypted Comparison in Wireless Sensor Networks (WSNs)

The idea of OPES [8] is to take as input a user provided target distribution T determined by the network designer and transform the plaintext value such that the transformed value follows the target distribution. In paper [5] the authors show how OPES can be adapted to the WSN.

As given in [5, 8], OPES have the following stages:

1. Model: The input distribution P and the target distribution T are modeled piecewise linear splines.
2. Flatten: The input distribution P is transformed into flat distribution F such that values in F are uniformly distributed.
3. Transform: The flat distribution F is transformed into cipher distribution C such that values in C are according to the target distribution T.

Let us look into the details of these phases as described in [5]. In the *Modeling phase* the sorted points $p_1<p_2<..p_{|p|}$ (samples sensed by the sensor node known to the network designer) are split into number of bucket, each bucket has boundaries $[p_l,p_h]$, p_l being the least value and p_h being the highest value. A given bucket $[p_l,p_h]$ has $h\text{-}l\text{-}1$ sorted points. The bucket is then split at the point that has the largest deviation from it's expected value. The splitting is then stopped when the number of points in the bucket is below some threshold. Using Minimum Description Length principle [15] the buckets can be minimized even while the values in the bucket preserve the uniform distribution. The bucket boundaries are uploaded onto each sensor. For m buckets the sensors stores $m+1$ bucket boundaries.

In the *Flattening phase* a plaintext bucket B is mapped onto a bucket B_f such that the density of the flattened bucket is uniform. If a distribution over $[0, ph]$ has a density function $qp+r$, where $p \in [0,ph]$, then for any constant $z>0$, the mapping function $M(p)$ will yield a uniform distribution. $M(p)$ is calculated as, $M(p) =z(qp^2/2r+p)$. $s=q/2r$ is called quadratic coefficient and during predeployment phase one coefficient for each bucket is uploaded to all sensor nodes. $z = Kn/(sw^2 + w)$ is a scale factor where w is the width of the bucket, n is the number of points in the bucket and K is the maximum of minimum of the predicted flattened bucket widths. (p_{min} ,p_{max}) represents the domain of the sensed valued in plaintext and (f_{min} ,f_{max}) is the domain of the sensed values. When sensor senses a plain text value the sensor node performs binary search over $m+1$ bucket boundaries. Then p is mapped on to flat value f using the equation,

$$f = f_{\min} + \sum_{j=1}^{i-1} w_j^f + M_i(p - p_{\min} - \sum_{j=1}^{i-1} w_j),$$

where $w_i^f = M_i(w_i)$. w_i is the length of the plaintext bucket B_i and w_i^f is the length of the corresponding flat bucket. The sensor stores the bucket boundaries, quadratic coefficient and scale factors in the data structure k^f, which is termed as encryption key used to flatten the sensed values.

In the *Transformation phase* the uniform flattened value is mapped into target distribution. In other words, the target distribution is flattened and aligned with the flattened plain text distribution. The sink node models the target distribution and flattens it during the predistribution phase. The modeling of the target distribution yields a set of buckets, $(B_1^t, B_2^t,..., B_k^t)$ and for each bucket there is a quadratic factor s^t and a scale factor z^t given as, $z^t = K_t n_t/(s^t (w_t^2)+w_t)$. The quadratic function and the scale factor is precomputed. Let B'^f be the bucket in the flattened target distribution with length w'^f. To align the flattened plain text distribution and the flattened target distribution, a scaling factor L is computed as, $L = \sum_{i=1}^{m} w_i^f \Big/ \sum_{i=1}^{k} w_i^{'f}$. The length of the cipher bucket B^c corresponding to the target bucket B^t is given as $w_i^c = L\, w_i^t$ and the length of the flattened target bucket w'^f is given as $w'^f = L\, w'^f$. Finally the mapping function M^c for mapping values from the bucket B^c to the flat bucket B'^f is defined by the quadratic coefficient $s^c = s^t/L$ and the scale factor $z_c = z_t$. If $[c_{min}, c_{max}]$ is the domain of the ciphertexts, then a flat value f from the bucket B'^f is mapped into cipher c using the equation

$$c = c_{\min} + \sum_{j=1}^{i-1} w_j^c + M_i^{c-1}(f - f_{\min} - \sum_{j=1}^{i-1} w_j^{'f})$$

The OPES scheme, when adapted for encrypted comparison in WSN is reasonably energy wise, as computationally intensive operation is performed at the sink node during predeployment stage. The sensor node performs minimal computation in real time. The sink node models the plaintext distribution, target distribution, and computes the scale factor and quadratic coefficient. Computation at the sensor nodes includes binary search of the sorted bucket boundaries, mapping plaintext value to flattened value and mapping flat value to cipher value [5]. This scheme is used to perform minimum/maximum aggregation function at the aggregator node.

In the next section we show how minimum/maximum can be performed at the aggregator node using homomorphic encryption schemes described in section 2.

4 Calculation of Aggregation function Maximum/Minimum

In this section we look at how to determine the aggregation function minimum/maximum by computing addition operation on the encrypted data at the aggregator node. The data is encrypted by the homomorphic encryption schemes mentioned in section 2.

4.1 Finding the maximum value at the aggregator node

To calculate the maximum value we use the scheme proposed by the author in [16]. The protocol chooses a weight w such that $(1<=w<=n)$ and chooses n such that it is large enough to represent the longest path. The weight w is encrypted as:

$e(w) = (e_1, e_2,... e_n)$

$$= \underbrace{E(z),...,E(z),}_{w} \underbrace{E(0),...,E(0)}_{n-w} \quad ...(1)$$

Here $e(w)$ is the encryption of weight w, $E(0)$ is the encryption of 0, $E(z)$ is the encryption of z and z is a number not equal to 0.

To use this protocol in WSN, we assume that the network designer chooses the value n, large enough to represent the maximum sensed value by the sensor nodes. The weight w is the data sensed by the sensor nodes. The network designer depending on the homomorphic encryption schemes chooses the value z. For *cryptosystem using mod operation* [9] and *Domingo-Ferrer's Privacy Homomorphism allowing field operation on encrypted data* [13] the value of z is chosen such that, $z*s<p$ and $z\neq 0$. Here s is the number of sensor nodes in WSNs and $s<p$. For *Domingo-Ferrer's New Privacy homomorphism* [10] the value of z is chosen such that $z<q$ and $z*(p-1) \bmod m \neq 0$. The number of sensor nodes s in WSNs should be lesser than p. In *Domingo-Ferrer's Additive and Multiplicative Privacy homomorphism* [6] the value z is chosen such that, $z*s<m'$ and $z\neq 0$. s is again the number of sensor nodes in WSNs and should be lesser than m'. The value of z is chosen with such restriction so that they do not add up to a value 0 at the aggregator nodes in WSNs.

The sensor nodes encrypt the sensed value as show in equation (1) and each sensor node transmits n encrypted data to the aggregator node.

The aggregator node calculates the maximum value by computing

$$M = \sum_{i=1}^{i=s}(E_{i,1}(x_1),...,E_{i,j}(x_j),...,E_{i,n}(x_n)) \quad ..(2)$$

on the encrypted data received by all the sensor nodes. Here s is the number of sensor nodes in the network, sending data to the aggregator node and $E_{i,j}(x_j)$ is the encryption of either $z \neq 0$ or 0. The aggregator node transmits the calculated maximum value $M = E_0(x), E_1(x), \ldots, E_n(x)$ to the sink node.

The sink node decrypts the maximum value from e_n to e_1 for $i=n$ to 1 until $D(E(x_i)) \neq 0$ and i determines the maximum value sensed by the sensor nodes.

Let us consider an example to understand this in more details. Assume that $n=5$ and there are 4 sensors (s_1, s_2, s_3, s_4) that monitor environmental data with readings $(1, 3, 4, 2)$ respectively.

The sensors encrypt the sensed data as

$s_1 : e(1) = E_{1,1}(z), E_{1,2}(0), E_{1,3}(0), E_{1,4}(0), E_{1,5}(0)$

$s_2 : e(3) = E_{2,1}(z), E_{2,2}(z), E_{2,3}(z), E_{2,4}(0), E_{2,5}(0)$

$s_3 : e(4) = E_{3,1}(z), E_{3,2}(z), E_{3,3}(z), E_{3,4}(z), E_{3,5}(0)$

$s_4 : e(2) = E_{4,1}(z), E_{4,2}(z), E_{4,3}(0), E_{4,4}(0), E_{4,5}(0)$

z is any value not equal to 0.

The sensor node then transmits these 4 encrypted data to the aggregator node.

The aggregator node computes $e(1)+e(3)+e(4)+e(2)$ to get $E(z),E(z),E(z),E(z),E(0)$. This is the maximum value sensed by the sensor node. Aggregator node transmits the maximum value to the sink node.

Sink node decrypts the received encrypted message $E(z),E(z),E(z),E(z),E(0)$ from right to left for $i=5$ to 1. At $i=4$ the encrypted message decrypts to a value z not equal to 0. So the maximum value is 4.

Let us look into a numerical example using Domingo-Ferrer's Privacy Homomorphism [6]. Let $d=2$, $m=28$, $r=3$ and $m' = 14$. Let $(x_1,x_2,x_3,x_4,x_5) = (1,2,3,0,0)$.

$E_k(x_1) = E_k(1) = E_k(10,5) = (2,17)$

$E_k(x_2) = E_k(2) = E_k(11,5) = (5,17)$

$E_k(x_3) = E_k(3) = E_k(5,12) = (15,24)$

$E_k(x_4) = E_k(0) = E_k(4,-4) = (12,20)$

$E_k(x_5) = E_k(0) = E_k(2,-2) = (6,10)$

Adding with encryption of 0 can further hide encryption of x, but the result is still x. $E_k(x_5) + E_k(x_6) = (18,2)$.

As before assume that $n=5$ and the 4 sensors (s_1, s_2, s_3, s_4) that monitor environmental data senses data as $(1, 3, 4\ 2)$ respectively. The data is encrypted as in equation (1), with the values encrypted with Domingo-Ferrer's Privacy Homomorphism

$s_1 : e(1) = E_k(3), E_k(0), E_k(0), E_k(0), E_k(0)$
$= (15,24),(12,20),(6,10),(12,20),(18,2)$

$s_2 : e(3) = E_k(1), E_k(3), E_k(2), E_k(0), E_k(0)$
$= (2,17),(15,24),(5,17),(12,20),(6,10)$

$s_3 : e(4) = E_k(2), E_k(1), E_k(3), E_k(1), E_k(0)$
$= (5,17),(2,17),(15,24),(2,17),(6,10)$

$s_4 : e(2) = E_k(3), E_k(1), E_k(0), E_k(0), E_k(0)$
$= (15,24),(2,17),(12,20),(6,10),(18,2)$

These encrypted values are sent to the aggregator node by the sensor node.

The aggregator node calculates the maximum value by computing,

$M = s_1 + s_2 + s_3 + s_4 = (15+2+5+15 \bmod 28,\ 24+17+17+24 \bmod 28),\ (12+15+2+2 \bmod 28,\ 20+24+17+17 \bmod 28),\ (6+5+15+12 \bmod 28,\ 10+17+24+20 \bmod 28),\ (12+12+2+6 \bmod 28,\ 20+20+17+10 \bmod 28),\ (18+6+6+18 \bmod 28,\ 2+10+10+2 \bmod 28)$

$M = (9,26),(3,22),(10,15),(4,11),(20,24)$

This maximum value M is transmitted to the sink node. The sink node decrypts the maximum value M from right to left. At $i=5$ the value $(20, 24)$ decrypts to 0, at $i=4$ the value $(4,11)$ decrypts to 1. Since at $i=4$ the encrypted value decrypts to a value $z \neq 0$, the maximum value is 4.

4.2 Finding the minimum value at the aggregator node

To calculate the minimum value we use the scheme proposed by the author in finding the minimum path [17], which modifies the proposed schemes in [16] which determines the maximum path. The protocol chooses a weight w such that $(1 <= w <= n)$ and chooses n such that it is large enough to represent the longest path. The weight w is encrypted as:

$e(w) = (e_1, e_2, \ldots e_n)$

$$= \underbrace{E(0), \ldots, E(0)}_{w},\ \underbrace{E(z), \ldots, E(z)}_{n-w} \quad \ldots (3)$$

Here $e(w)$ is the encryption of weight w, $E(0)$ is the encryption of 0, $E(z)$ is the encryption of z and z is a number not equal to 0.

To use this protocol in WSN the adaptation is same as in the earlier section. The network designer chooses the value n, large enough to represent the largest sensed data and the weight w is the data sensed by the sensor node. As in the earlier section the network designer depending on the homomorphic encryption schemes chooses the value z. For *cryptosystem using mod operation* [9] and *Domingo-Ferrer's Privacy Homomorphism allowing field operation on encrypted data* [13] the value of z is chosen such that, $z*s<p$ and $z \neq 0$. Here s is the number of sensor nodes in WSNs and $s<p$. For *Domingo-Ferrer's New Privacy homomorphism* [10] the value of z is chosen such that $z<q$ and $z*(p-1) \bmod m \neq 0$. The number of sensor nodes s in WSNs should be lesser than p. In *Domingo-Ferrer's Additive and Multiplicative Privacy homomorphism* [6] the value z is chosen such that, $z*s<m'$ and $z \neq 0$. s is again the number of sensor nodes in WSNs and should be lesser than m'. The value of z is chosen with such restriction so that they do not add up to a value 0 at the aggregator nodes in WSNs.

The sensor nodes encrypt the sensed value as show in equation *(3)*, and transmit the encrypted data to the aggregator node. The aggregator node calculates the minimum value by adding up the encrypted data received from all the sensor nodes as

$$M = \sum_{i=1}^{i=s} (E_{i,1}(x_1),...,E_{i,j}(x_j),...,E_{i,n}(x_n)) \quad ..(4)$$

Here s is the number of sensor nodes in the network, sending data to the aggregator node and $E_{i,j}(x_j)$ is the encryption of either $z \neq 0$ or 0.The aggregator node transmits the calculated minimum value M to the sink node.

The sink node decrypts the minimum value from e_1 to e_n for $i=1$ to n until $D(E(x_i))$ not equal to 0 and $i-1$ determines the minimum value sensed by the sensor node.

As in the earlier section, let us consider an example to understand this in more details. Assume that $n=5$ and there are 4 sensors (s_1, s_2, s_3, s_4) that monitor environmental data and they measure sensor readings $(1,3,4,2)$ respectively. The sensor nodes encrypt the sensed data as

$s_1 : e(1)=E_{1,1}(0), E_{1,2}(z), E_{1,3}(z),E_{1,4}(z), E_{1,5}(z)$

$s_2 :e(3) = E_{2,1}(0), E_{2,2}(0), E_{2,3}(0),E_{2,4}(z), E_{2,5}(z)$

$s_3 :e(4) = E_{3,1}(0), E_{3,2}(0), E_{3,3}(0),E_{3,4}(0), E_{3,5}(z)$

$s_4 :e(2) = E_{4,1}(0), E_{4,2}(0), E_{4,3}(z),E_{4,4}(z), E_{4,5}(z)$

z is any value not equal to 0. The sensor node then transmits these 4 encrypted data to the aggregator node.

The aggregator node computes $e(1)+e(3)+e(4)+e(2)$ to get $E(0), E(z), E(z), E(z), E(z)$. This is the minimum value sensed by the sensor nodes. Aggregator node transmits the minimum value to the sink node.

Sink node decrypts the received encrypted message $E(0),E(z),E(z),E(z),E(z)$ from left to right for $i=1$ to $n=5$. At $i=2$ the encrypted message decrypts to a value $z \neq 0$. So the minimum value is $i-1 = 1$.

Let us look into a numerical example as before using Domingo-Ferrer's Privacy Homomorphism [6]. Let $d=2$, $m=28$, $r=3$ and $m' = 7$. Let $(x_1,x_2,x_3,x_4,x_5) = (1,2,3,0,0)$.

$E_k(x_1) = E_k(1) = E_k(10,5) = (2,17)$

$E_k(x_2) = E_k(2) = E_k(11,5) = (5,17)$

$E_k(x_3) = E_k(3) = E_k(5,12) = (15,24)$

$E_k(x_4) = E_k(0) = E_k(4,-4) = (12,20)$

$E_k(x_5) = E_k(0) = E_k(2,-2) = (6,10)$

Adding with encryption of 0 can further hide encryption of x, but the result is still x. $E_k(x_5)+ E_k(x_6)=(18,2)$.

As before assume that $n=5$ and the 4 sensors $(s1, s2, s3, s4)$ that monitor environmental data senses data as $(1, 3, 4, 2)$ respectively. The data is encrypted as in equation (1), with the values encrypted with Domingo-Ferrer's Privacy Homomorphism

$s_1 : e(1)= E_k(0),E_k(3), E_k(1), E_k(2), E_k(3)$
$=(6,10),(15,24),(2,17),(5,17),(15,24)$

$s_2 :e(3) = E_k(0),E_k(0), E_k(0), E_k(1), E_k(3)$
$=(12,20),(6,10),(18,2),(2,17),(15,24)$

$s_3 :e(4) = E_k(0),E_k(0), E_k(0), E_k(0), E_k(1)$
$=(18,2),(12,20),(6,10),(12,20),(2,17)$

$s_4 :e(2) = E_k(0),E_k(0), E_k(3), E_k(1), E_k(2)$
$= (12, 20),(18,2),(15,24),(2,17),(5,17)$

These encrypted values are sent to the aggregator node by the sensor nodes.

The aggregator node calculates the minimum value by computing,

$M = s_1+ s_2+ s_3+ s_4 = (6+12+18+12 \bmod 28, 10+20+2+20 \bmod 28), (15+6+12+18 \bmod 28, 24+10+20+2 \bmod 28), (2+18+6+15 \bmod 28, 17+2+10+24 \bmod 28), (5+2+12+2 \bmod 28, 17+17+20+17 \bmod 28), (15+15+2+5 \bmod 28, 24+24+17+17 \bmod 28)$

$M = (20, 24),(23,0),(13,25),(21,15),(9,26)$

This minimum value M is transmitted to the sink node. The sink node decrypts the minimum value M from left to right. At $i=1$ the value $(20, 24)$ decrypts to 0, at $i=2$ the value $(23,0)$ decrypts to 3. Since at $i=2$ the encrypted value decrypts to a value $z \neq 0$, the minimum value is $i-1=1$.

The network administrator can place the encrypted values $E(z)$, $E(0)$ at each sensor node during the predeployment stage of the sensor nodes. The network administrator can also determine the number of encrypted values to be stored at the senor nodes depending upon the storage limitation of the sensor nodes. The sensor node can randomize the values of *E(z) and E(0)*, by computing addition operation with $E(0)$. The sensor node has low computation power and encryption at the sensor node in WSNs is a very costly operation. By using these proposed schemes the sensor nodes need not encrypt any data and hence remove the computation cost of encryption all together. Sensor nodes need not store the encryption key and even if the node is tampered with, the key won't be revealed. The sensor nodes after sensing the data transmit n encrypted values. The aggregator node performs minimum/maximum value by computing at most ns addition operations resulting in n encrypted values. Here n is the number of encrypted values sent by each sensor node and s is the number of sensor nodes.

We have used privacy homomorphism as an example, but since they are vulnerable to known plaintext attacks it might be a problem. We can use any additive homomorphic encryption schemes, which is secure against known plaintext attacks. Okamota and Uchiyama's new public-key cryptosystem[18], Paillier three new probabilistic encyption scheme[19], Elliptic curve ElGamal encryption scheme[20] are some of the additive homomorphic encryption scheme secure against known plaintext attacks. The encryption cost is not a problem as the encryption is done during the predeployment stage of the sensor nodes.

5 Conclusion

In this paper we have shown that the aggregator node can perform aggregation function maximum/minimum by computing addition operation and not comparison operation on the data encrypted using homomorphic encryption schemes. By pre-computing *E(z), E(0)* we have eliminated the computation cost for encryption at the sensor nodes and solved the major problem in WSNs. By using our scheme one can use any additive homomorphic encryption schemes, as encryption cost at the sensor node in WSNs is not a problem. Furthermore, by performing addition operation over encrypted data to calculate minimum/maximum we have eliminated the overhead of OPES required while calculating minimum/maximum.

6 REFERENCES

1. Yang Xiao. "Security in Sensor Networks", Auerbach Publications, 2007, pp. 275-290.
2. Dirk Westhoff, Joao Girao, Mithun Acharya. "Concealed data aggregation for reverse multicast traffic in sensor networks: Encryption, key distribution and routing adaptation". IEEE Transactions on Mobile Computing,, October 2006.
3. J.Girao, D.Westhoff, and M.Schneider. Concealed data aggregation in wireless sensor networks. ACM WiSe04 – poster, in conjunction with ACM MOBICOM 2004, October 2004.
4. J.Girao, D.Westhoff, and M.Schneider. CDA: Concealed data aggregation for reverse multicast traffic in wireless sensor networks. 40th International conference on communications, IEEE ICC 2005, May 2005.
5. Mithun Acharya, Joao Girao, and Dirk Westhoff. "Secure comparison of encrypted data in wireless sensor networks". In *3rd Intl. Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, Trentino, Italy, April 2005. WiOpt2005
6. J. Domingo-Ferrer. "A Provably Secure Additive and Multiplicative Privacy Homomorphism". Information Security Conference, LNCS 2433, pages 471–483, January 2002.
7. R. L. Rivest, L. Adleman and M. L. Dertouzos, "On data banks and privacy homomorphisms", in *Foundations of Secure Computation*, R. A. DeMillo *et al.*, Eds. New-York: Academic Press, 1978, pp. 169-179.
8. Rakesh Agrawal., Jerry Kiernan, Ramakrishnan Srikant, Yirong Xu: "Order-Preserving Encryption for Numeric Data". SIGMOD Conference 2004: 563-574.
9. Hyungjick Lee, Jim Alves-Foss,Scott Harrison, " The use of Encrypted Functions for Mobile Agent Security",Proceedings of the 37th Hawaii International Conference on System Sciences – 2004.
10. Josep Domingo i Ferrer, "A new Privacy Homomorphism and Applications", Elsevier North-Holland, Inc, 1996.
11. William Stallings "Cryptography and Network Security", Third Edition, Chinese Remainder Theorem (CRT), pp. 245-47.
12. Jung Hee Cheon, Hyun Soon Nam,"A Cryptanalysis of the Original Domingo-Ferrer's Algebraic Privacy Homorphism", http://eprint.iacr.org/2003/221.pdf
13. J. Domingo-Ferrer and J. Herrera-Joancomarti. "A privacy homomorphism allowing field operations on encrypted data". I Jornades de Matematica Discreta i Algorismica, Universitat Politecnica de Catalunya, March 1998.
14. D. Wagner, "Cryptanalysis of an algebraic privacy homomorphism", *In proceedings of the 6th information security conference(ISC03)*, Bristol, UK, October 2003.
15. J.Rissanen. "Stochastic complexity in statistical inquiry". World Scientific Publication, 1989.
16. Makoto Yokoo, Koutarou Suzuki, "Secure Multi-agent Dynamic Programming based on Homomorphic Encryption and its Application to Combinatorial Auctions", Proceedings of the First International joint Conference on Autonomous Agents and Multiagent systems(AAMAS), 2002.
17. L.Ertaul, Vaidehi, "Finding Minimum Optimal Path Securely Using Homomorphic Encryption Schemes in Computer Networks**"**, The 2006 International Conference on Security & Management, SAM'06**,** June, Las Vegas.
18. T. Okamoto and S. Uchiyama. "A New Public-Key Cryptosystem as Secure as Factoring". *EUROCRYPT*, pages 308–318, 1998.
19. P. Paillier. "Trapdooring Discrete Logarithms on Elliptic Curves over Rings". *ASIACRYPT*, pages 573–584, 2000.
20. Willian Stallings,"Cryptography and Network Security, Principles and Practices."Fourth Edition, Prentice Hall 2006, pp.312.