A Secure Scheme for Aggregating Encrypted Data Against Outsider Attacks in Wireless Sensor Networks

Mitra Baratchi, Kamal Jamshidi
Department of Computer Engineering
University of Isfahan
Isfahan, Iran
{Mitra.Baratchi, Jamshidi@eng.ui.ac.ir}

***Abstract*— Wireless sensor networks are low powered energy operated sensors mainly deployed in remote environments. Using methods to save energy is of utmost importance in these systems. Data transmission in such networks is by far the most energy consuming task and by reducing its contribution a great amount of energy can be saved. Data aggregation can extend network lifetime and reduce its traffic. However, providing security requirements such as, end-to-end confidentiality or aggregate integrity while performing data aggregation is a challenging task. Here, we propose a scalable method for aggregating encrypted data. The integrity of aggregated data from outsider attacks is checked through a distributed method. Compared to the previous method, our scheme achieves better performance in presence of silent nodes and avoids blind rejection in presence of outsider attacks.**

Keywords-security; encrypted data aggregation; aggregate integrity; wireless sensor networks

I. Introduction

Recently, research in the field of wireless sensor networks has become increasingly popular due to its vast potential domain of applications such as military surveillance, and habitat monitoring. Wireless sensor networks are composed of a large number of low-powered and battery operated sensors with limited computation, communication, and power resources. In these networks nodes collaborate to gather measurements from an area and route them to a processing center. A major challenge in such networks is energy efficiency, because nodes are low powered and battery operated. Therefore, all protocols should be designed by considering solutions for saving energy [1]. Data aggregation is a major solution to this challenge. By combining and summarizing data from different sensor nodes, data aggregation reduces the amount of data transmission and increases network lifetime. Data aggregation can be performed by using any algebraic function with multiple inputs and one output, such as addition and multiplication or by any statistical operation on the numerical measurements, such as median, minimum and maximum. Although data aggregation improves data bandwidth and energy utilization, it adversely affects other performance metrics such as security. Aggregated data are prone to different malicious adversary attacks [2]. Many of the applications of wireless sensor networks are security critical, thus providing the security of aggregated data in such networks is of utmost importance. Accordingly, designing aggregation methods with essential security requirements such as, end-to-end confidentiality, integrity, authentication, and availability has become an active field of research.

End-to-end confidentiality ensures that the secrecy of data is not disclosed to either authorized or unauthorized parties in the network. This secrecy is provided through a concept called concealed data aggregation. Alternatively, aggregate integrity ensures that the aggregated data has not been tampered en-route by unauthorized parties. Although it is possible to provide end-to-end confidentiality, it is impossible to provide end-to-end integrity, because the aggregation function itself changes the original data. There have been some constructions to provide aggregate integrity [3-6]. In most of the proposed methods, the aggregated data is verified through a call back from the Base or aggregator nodes which imposes further data transmission on the network. Furthermore, mainly the verification is done in a centralized manner at the Base and no verification is done prior to the arrival of the result at the Base. One major drawback of this style of verification is that a single attack on a sensor reading will lead to rejection of all valid readings which have contributed to the aggregation procedure and have been routed to the Base. In this paper, we propose a scheme for aggregating encrypted data based on bihomomorphic encryption. We enhance the method proposed in [7], which provides end-to-end confidentiality of aggregated data using a distributed verification scheme. In our scheme, outsider attacks are diagnosed at each intermediate aggregator and their impact on final aggregation is neutralized, avoiding blind rejection of valid data. Moreover, this method is efficiently scalable in terms of silent aggregators and sensor nodes, and provides availability by replacing missing values. A coordinated scheme is used to update all encryption keys and necessary values locally at each session. Evaluation results show that our method performs efficiently at presence of silent nodes.

II. RELATED WORKS

Early works in data aggregation considered every mote to be honest [8, 9]. Because of the necessity of secure aggregation methods, researchers have worked on secure data aggregation in a hop-by-hop manner [3-5, 11]. In this group of protocols, generally, sensor nodes encrypt their sensed messages and send them to the aggregators. The aggregators need to decrypt the messages to be able to perform data aggregation and encrypt them before forwarding. A method for detection of node misbehaviors through delayed aggregation is proposed in [3]. This protocol ensured integrity through a key chain and authentication keys which were periodically broadcasted by the Base station. However, it did not address confidentiality. Different methods have been proposed to check the correctness of aggregated data including using random sampling and interactive proofs [4] through a Merkle HashTree [10], or collecting information from witness nodes [5]. Witness nodes are the nodes that perform data aggregation and compute MACs (Message authentication code) but only send the MACs to the base station. These MACs are used for verification purposes at the Base Station. A secure hop-by-hop method based on divide-and-conquer and commit-and-attest is proposed in [11], where the trust on high level aggregators is reduced to alleviate the potential security risk caused by compromised high level nodes. The use of traditional cryptography methods in above algorithms did not allow achieving end-to-end confidentiality and all these works could only provide confidentiality in a hop-by-hop manner through decryption/encryption at each intermediate aggregator. Furthermore, each intermediate aggregator needed to store many keys for decryption and encryption. Therefore, this routine caused both energy waste and security risks [12]. It also complicates key management, as nodes need to share keys with their neighbors.

To alleviate the drawbacks of hop-by-hop schemes a number of methods have been proposed, which perform data aggregation without any need to decrypt the received data. This procedure is not possible through traditional cryptography algorithms but it is possible through privacy homomorphism.

Privacy homomorphism is a transformation, which performs direct computation on encrypted data and allows end-to-end confidentiality between the sensors and the sink node[7, and 13-15]. The advantage of this method is that the need for encryption/decryption in intermediate aggregators is eliminated. Instead, the aggregators can simply aggregate the incoming messages and forward the result to the predecessor. Upon receiving the results, the Base Station will be able to decrypt it. No intermediate aggregator has the knowledge to decrypt the incoming ciphertext.

Additive and multiplicative homomorphic over data set Q method are formulized as follows, where K_{en} and K_{de} are encryption and decryption keys while D and E are decryption and encryption:

$$a + b = D_{K_{de}}\left(E_{K_{en}}(a) + E_{K_{en}}(b)\right) \text{ where } a \text{ and } b \in Q \tag{1}$$

$$a \times b = D_{K_{de}}\left(E_{K_{en}}(a) \times E_{K_{en}}(b)\right) \text{ where } a \text{ and } b \in Q \tag{2}$$

Generally, the privacy homomorphism methods can be divided into two categories: (i) those based on symmetric method[13,14,7], and (ii) those based on asymmetric method[15].

The concept of concealed data aggregation and aggregating encrypted data was first introduced in [13], where a symmetric additive/multiplicative privacy homomorphism method was used for the purpose of aggregating ciphertext rather than plaintext [16]. Although this method provided end-to-end data confidentiality, all the nodes and the Base Station share a single key and if one node is compromised the whole network will be disrupted.

Alternatively, an additive concealed data aggregation scheme is introduced in [14] using modular addition based on one-time pads. This method is further extended in [7] by an aggregate authentication scheme, which can recognize outsider attacks at the Base Station. The major drawback of this method is that each aggregator needs to append the list of non-contributing nodes to the message, thus when the number of non-contributing nodes is large a large overhead will be imposed on the forwarding aggregators.

Lately an aggregation method was proposed in [17] which could compare messages and eliminate redundant readings without decrypting them. Moreover, using public key based privacy homomorphism methods has been studied in [15], which showed that for the networks where energy is the most important factor, using symmetric key methods is more favorable.

A. *Concealed data aggregation*

As mentioned before, concealed data aggregation was first introduced in [12] to perform aggregation on encrypted data.

For encryption transformations $E : \mathbb{K} \times \mathbb{P} \rightarrow \mathbb{C}$, and for decryption transformations $D : \mathbb{K} \times \mathbb{C} \rightarrow \mathbb{P}$, where the plaintext, ciphertext, and key spaces are denoted by ($\mathbb{P}$, +), ($\mathbb{C}$, $\oplus$), and ($\mathbb{K}, \odot$). The concealed data aggregation (CDA) requires the existence of a subset $\mathbb{K}_\alpha \subseteq \mathbb{K}^\alpha$ and a function $K_\alpha : \mathbb{K}_\alpha \rightarrow \mathbb{K}$ such that for all $(k_1 \ldots . k_\alpha) \in \mathbb{K}_\alpha$ and $(v_1 \ldots . v_\alpha) \in \mathbb{P}^\alpha$, ($\alpha \geq 1$):

$$D_{K_\alpha(k_1 \ldots k_\alpha)}(\oplus_{i=1}^{\alpha} E_{k_i}(v_i)) = \sum_{i=1}^{\alpha} v_i \tag{3}$$

A bihomomorphic encryption transformation introduced in [18] is a transformation, in which the encryption is homomorphic both on the plaintext space and on the key space. It means that for all keys $k_1 \ldots k_\alpha \in \mathbb{K}$ and the key generator *kg* there is a key $k \in \mathbb{K}$ such that:

$$kg(k_1, t) \odot \ldots \odot kg\,(k_\alpha, t) = kg(k, t) \tag{4}$$

In this paper, we use modulo integer addition as a bihomomorphic transformation by replacing $\odot$ and $\oplus$ with modular addition. It means that for $\mathbb{K} = \mathbb{P} = \mathbb{Z}\ (\mathrm{mod}\ n)$ with an integer $n \geq 1$, $E_k(m) = k + m$ mod n is a bihomomorphic encryption.

III. Network and attacker model

A. *Network model*

We consider a large and densely deployed sensor network with |*N*| nodes, which is arranged in a topological tree. Three roles are defined in the network as sensor nodes, aggregators and the Base Station:

- **Sensor nodes:** which sense, encrypt and forward the encrypted data and its checksum to the aggregator.
- **Aggregator nodes:** which sense, encrypt, perform aggregation on encrypted data, and verify the correctness of the encrypted messages against their checksum and replace the missing values with the dummy ones.
- **Base Station:** which is able to decrypt the received encrypted messages and remove dummy values from the aggregation result.

Each node S_i is a wireless sensor node such as Mica2 motes. Due to key refreshments and encryption keys specific to each phase, our aggregation method requires that all sensor nodes send their data to the Base Station at the same period. Therefore, we assume that nodes are loosely synchronized with a constant time interval.

We assume that after deployment, nodes arrange with a spanning tree protocol and each node identifies its parent and child nodes. Each node performs only one measurement per period. In each epoch, some nodes might choose to remain silent due to energy saving purposes and their measurement is replaced by dummy values as described in [18]. The wireless communication channel is not fully reliable, and thus the messages might be lost or corrupted before reaching the destination.

Before deployment, each sensor node S_i is equipped with unique keys K_i (encryption key), K_i' (checksum key), unique coefficients, co_i ,co_i' , and a common group key *KG*. The group key is shared by all nodes. Each aggregator node will additionally be equipped with the aggregation of its sensor child nodes (direct or indirect) checksum key $\overline{K_a'} = \odot_{i \in succ(a)} K_i'$, dummy values $D_{i \in child(a)} = (\oplus_{n \in succ(i)}(R \oplus K_n))\oplus(R \oplus K_i)$ and dummy checksums $\underline{Dc_{i \in child(a)}} = \oplus_{n \in succ(i)}(KG.D_i \oplus K_n') \oplus K_i'$. It will also receive the aggregated coefficients $\overline{co'_{i \in child(a)}} = (\odot_{n \in succ(i)} co_n') \odot co_i'$, and $\overline{co_{i \in child(a)}} = (\odot_{n \in succ(i)} co_n) \odot co_i$ for each of its child nodes i. By *succ(a)* we mean all the offspring sensor nodes under the aggregator. For example, the sensor child nodes of the base are all the sensor nodes in the tree. *Child(a)* means only the direct children a and $|n_i|$ is the number of sensor nodes from i. The dummy values can be used to replace silent nodes measurements and will be later reduced from the final aggregation result at the base.

The Base Station stores KG as well as the master keys of all individual keys and master coefficients as below:

$$\widehat{K} = \odot_{i=1}^{N}\ K_i \tag{5}$$

$$\widehat{K}' = \odot_{i=1}^{N} K_i' \tag{6}$$

$$\widehat{co} = \odot_{i=1}^{N}\ co_i \tag{7}$$

$$\widehat{co}' = \odot_{i=1}^{N} co_i' \tag{8}$$

B. Updating keys

For security reasons, during each sampling phase all the keys and dummy values should be updated. These values should be updated in a manner that the sink node and intermediate aggregators are aware of the refreshment. We assume that during the setup phase each sensor node is equipped with a public pseudorandom function [19] ($F_\lambda = \{f_s : \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda\}\ s \in \{0,1\}^\lambda$), which will be used to produce unique encryption and checksum keys for each phase. There is no restriction in the choice of the pseudorandom function. Using this pseudorandom function each node produces an update value which will be used for computing private and global keys of each node and to update the dummy values relating to each child node. The update value is computed as follows:

$$\Delta^r = f_{KG}(r) \tag{9}$$

Accordingly, encryption keys of sensor $S_i$at round *r* can be computed as follows:

$$KG^r = KG.\Delta^r \tag{10}$$

$$K_i^r = K_i \odot co_i.\Delta^r \tag{11}$$

$$K_i'^r = K_i' \odot co_i'.\Delta^r \tag{12}$$

To make this updating scheme practical it is necessary that the intermediate aggregators are able to update the aggregation of checksum keys. For this purpose, it is necessary that each aggregator *a* computes that rounds addition values Δ^r and updates the aggregated checksums as follows:

$$\overline{K_a'^r} = (\odot_{i \in child(a)} \overline{co_a'}.\Delta^r\ ) \odot \overline{K_a'} \tag{13}$$

Respectively, the Base Station will update its master keys by:

$$\widehat{KG}^r = \widehat{KG}.\Delta^r \tag{14}$$

$$\widehat{K}^r = \widehat{K} \odot \widehat{co}.\Delta^r \tag{15}$$

$$\widehat{K}'^r = \widehat{K}' \odot \widehat{co}'.\Delta^r \tag{16}$$

C. Updating dummy values

In our method, we use dummy values to make the scheme scalable in terms of non-contributing nodes in each phase of data aggregation, and also to be able to replace messages which are invalidated in a distribution validation scheme. These dummy values can simply be any number such as *R,* which is encrypted with the session key of each sensor node only if the base is aware of the number. As we go up in the tree hierarchy, the dummy representator of each node replaces larger number of data. Generally, dummy values and checksums stored at aggregator *a* are defined as:

$$D_{i \in child(a)} = (R \odot K_{j \in succ(i)}) \oplus (R \oplus K_i) \tag{17}$$

$$Dc_{i \in child(a)} = KG.D_i \oplus (\odot K'_{j \in succ(i)}) \oplus K_i' \tag{18}$$

If for any reason a node's message was not delivered to the parent aggregator, a dummy value and checksum will replace it and will be finally reduced from the aggregation result at the Base. Since the dummy values replace the encrypted data and the encrypted data are computed with session updated keys, the dummy values and checksums need to be updated at each session in order to be valid. This updating is close to that of session keys. It can be done by adding the addition value to the previous session dummy value as follows:

$$D^r_{i \in child(a)} = D_i \odot (\overline{co_\iota}.\Delta^r) \tag{19}$$

Updating dummy checksums is not as easy as updating dummy values but can be done through the following operation:

$$Dc^r_{i \in child(a)} = (KG^r.D_i^r \oplus \overline{K_\iota'^r}) \oplus K_i'^r =$$
$$(\Delta^r.KG).(D_i \oplus ((\overline{co_\iota}).\Delta^r)) \oplus (\odot_{n \in succ(i)}(co_n'.\Delta^r \odot K_n') \odot (co_i'.\Delta^r \odot K_i'))) =$$
$$((\Delta^r + 1 - 1)).KG.(D_i \oplus ((\overline{co_\iota}).\Delta^r)) \odot ((\overline{co_i'}).\Delta^r) \odot (\odot_{n \in succ(i)} K_n') \oplus K_i' =$$
$$((\Delta^r - 1).KG.(D_i^r)) \oplus (KG.(D_i) \oplus (\odot_{n \in succ(i)} K_n') \oplus K_i') \oplus (KG.(\overline{co_\iota}.\Delta^r)) \oplus ((\overline{co_i'}).\Delta^r) =$$

$$Dc_i \oplus ((\Delta^r - 1).KG.(D_i^r)) \oplus (\Delta^r.(KG.\overline{co_i} \oplus \overline{co_i'})) \quad (20)$$

D. model

Here, we discuss the attack model on the model network depending on the adversaries' abilities:

- We assume a global attacker, who is capable of eavesdropping communication between sensor nodes without compromising any sensor node.
- The adversary knows everything about the system such as network structure, the key distribution, and deployed mechanism except for the secret keys.
- The adversary is capable of fully controlling the communication channel. She can perform attacks by catching, eavesdropping, destroying and replaying messages.
- The attacker is capable of eavesdropping, altering and stopping outgoing messages of each node, but not all nodes at the same time.

IV. Proposed scheme

Our work is motivated by the work reported in [7], which is called CMT method after the name of authors. In CMT method, the encrypted data are aggregated using privacy homomorphism, in which aggregate authenticates can be used against outsider only attacks. In this scheme, ciphertext c_i and its checksum cs_i are computed as follows:

$$c_i = m_i \oplus K_i^r \quad (21)$$

$$cs_i = m_i.KG^r \oplus K_i'^r \quad (22)$$

Where $m_i \epsilon [0, M\text{-}1]$ is the message, $\oplus$ is modulo M operator, KG^r is common group key, and K_i^r , $K_i'^r$ are encryption and checksum keys of node S_i in session r. The aggregation result and its checksums were computed through:

$$c^r = \oplus_{i\,\epsilon\, contributors} c_i \quad (23)$$

$$cs^r = \oplus_{i\,\epsilon\, contributors} cs_i \quad (24)$$

The aggregated result was retrieved at the Base through:

$$m^r = c^r - (\oplus_{i\,\epsilon\, contributors} K_i^r) \bmod M \quad (25)$$

And the validity of encrypted messages with their checksum was verified at the Base Station through checking:

$$cs^r ?= m^r.KG^r \oplus (\oplus_{i\,\epsilon\, contributors} K_i^r) \quad (26)$$

However, in this method the result of false aggregate is only found and rejected at the Base Station and no verification method is used before the arrival of aggregated data at the Base. Only a single malicious node can cause blind rejection, and thus an important amount of correct data is lost.

In our proposal we introduce a method, which allows middle way aggregation and verification of the encrypted data. First, we will describe a top level view of our algorithm, then we will present the algorithm executed at the aggregators and the sensor nodes.

In each sampling period, each sensor node encrypts its sensed measurement and computes its checksum. For the purpose of authentication, we use a method close to what proposed in CMT with some differences. Instead of computing the checksums from the plaintext, we compute aggregate checksums from the encrypted messages. This change will provide us the ability of middle way validation.

This encrypted message and its checksum are sent to the aggregator node by each child node. Upon receiving the sensed measures and their checksums, the aggregator node will aggregate encrypted messages, their checksums, and dummy counters. If any child node did not send its message a dummy value and checksum will replace its message and the number of sensor nodes from this child is added to the dummy counter. For verification purpose, each aggregator node has the group key and the sum of private checksum keys used by its children. Having this information, the aggregator can validate the aggregated messages and their checksums. Invalid aggregate checksum notifies that there has been an outsider attack in the way from at least one of its child nodes. To inactive this attack, the aggregator node will disregard all the messages it has received from its children, and compute an aggregate value and checksum from the encrypted dummy values of each node, representing the destroyed values. As the level of the aggregator is closer to the sink, the dummy values replace larger number of leaf nodes measurements and this number is set as the dummy counter. Finally, the result of aggregation will be aggregated with the aggregator's encrypted value, and will be sent to the upper level aggregator along with the dummy counter. These dummy values will be removed from the final result at the Base Station. Instead of validating the aggregated value, the aggregator can validate each node's message as well. Of course for the purpose of individual validation, the aggregator needs to store the aggregated checksum keys of each child node, which reduces security robustness. To minimize the computation at each aggregator node, we preferred to validate the aggregated result. But to prevent the outsider attacks which happen perpetually, the better alternative is to validate each node's message and checksum individually and expel the attacked nodes from the tree with the help of a reputation function. The dummy values also can be used to replace messages from the silent nodes so that the Base Station would not need the node ids for the purpose of decryption.

In the following section we try to elaborate the details of the protocol through the algorithms executed at nodes and aggregators. In these algorithms, *ctr* denotes dummy counter, $|n_i|$ is the number of successor sensors from i (including *i*), $a \oplus$ b denotes (*a*+*b* mod *M*), and *M* is sufficiently large such that if *N* different ciphers are added their sum is still smaller than *M*, ($\sum_{i=1}^{N} m_i \leq M$).

A. The algorithm executed by leaf node i at time frame r:

```
1) Compute addition value of time frame r: $\Delta^r = f_{KG}(r)$;
2) Compute $KG^r, K_i^r, K_i'^r$;
3) Sense plaintext message $m_i \in [0, M\text{-}1]$ ;
4) Encrypt message $m_i$ : $c_i = m_i \oplus K_i{}^r$;
5) Compute checksum: $cs_i = (KG^r.c_i) \oplus K_i{}'^r$;
6) Send $(c_i, cs_i)$ to the parent node;
```

B. The algorithm executed by aggregator a at time frame r:

```
1) Compute addition value of time frame r: $\Delta^r = f_{KG}(r)$
2) Compute $KG^r, \overline{K_a'^r}, D^r_{i \in child(a)}, Dc^r_{i \in child(a]}$;
3) Sense plaintext message $m_a \in [0, M\text{-}1]$ ;
4) Encrypt message $m_a$ : $c_a = m_a \oplus K_i{}^r$;
5) Compute checksum: $cs_a = (KG^r.c_a) \oplus K_a{}'^r$;
6) Replace a dummy ciphertext and a dummy checksum for each node $i$ the message of which is not arrived: $c_i = D_i^r$, $cs_i = Dc_i^r$, and $ctr_i = |n_i|$ ;
7) Compute aggregate ciphertext: $c_{agg} = \oplus_{i \in child(a)} c_i$ ;
8) Compute aggregate checksum: $cs_{agg} = \oplus_{i \in child(a)} cs_i$;
9) Update dummy counter: $ctr = \sum_{i \in child(a)} ctr_i$;
10) Validate aggregated message: if $(cs_{agg} == KG^r.c_{agg} \oplus \overline{K_a'^r})$ set $v=1$, else set $v = 0$;
11) If $v = 0$, compute the dummy aggregate and checksum of all child nodes: $c_{agg} = \oplus_{i \in child(a)} D_i^r$, $cs_{agg} = \oplus_{i \in child(a)} Dc_i^r$, and $ctr = |n_a|$;
12) Aggregate the result with a's individual ciphertext and checksum: $c_{agg} = c_{agg} \oplus c_a, cs_{agg} = cs_{agg} \oplus cs_a$ ;
13) Send $(c_{agg}, cs_{agg}, ctr)$ to the parent node;
```

V. EVALUATION

A. Security

1) Ciphertext only attacks: In these types of attacks, the only information that can be possessed by the adversary is the ciphertext of some nodes. Using session keys, double encryption of a message will not result in the same ciphertext. Therefore, the adversary cannot guess the equality of messages.

By eavesdropping the messages the adversary can form the following equations on checksum values:

$$Cs_1^{s1} = (KG)(\Delta^{s1})(C_1^{s1}) \oplus (K_1' \odot \Delta^{s1}.co_1') \qquad Cs_1^{s2} = (KG)(\Delta^{s2})(C_1^{s2}) \oplus (K_1' \odot \Delta^{s2}.co_1') \qquad ....$$
$$.... \qquad .... \qquad ....$$
$$Cs_N^{s1} = (KG)(\Delta^{s1})(C_N^{s1}) \oplus (K_N' \odot \Delta^{s1}.co_N') \qquad Cs_N^{s2} = (KG)(\Delta^{s2})(C_N^{s2}) \oplus (K_N' \odot \Delta^{s2}.co_N') \qquad ....$$

If the above equations were ordinary equations, the adversary could find the variables after few sessions by eavesdropping a number of nodes. However, since these equations are non-linear modular equations, the adversary is not able to achieve a single answer. The adversary will be able to assume that these equations are non-modular and solve them. For this purpose she should know what coefficient of *M* has been removed from each equation. Therefore she can form the following non-modular equations:

$$Cs_1^{s1} = (KG)(\Delta^{s1})(C_1^{s1}) + K_1' + (\Delta^{s1}.co_1') - x_1.M \qquad Cs_1^{s2} = (KG)(\Delta^{s2})(C_1^{s2}) + K_1' + (\Delta^{s2}.co_1') - x_2.M \qquad ....$$

....

Cs_N^{s1}= $(KG)(\Delta^{s1})(C_N^{s1})$+ $K'_N + (\Delta^{s1}.co'_N)$ - $x_n.M$ Cs_N^{s2}= $(KG)(\Delta^{s2})(C_N^{s2})$ +$K'_N + (\Delta^{s2}.co'_N)$ - $x_{n+1}.M$ ....

As we can see each equation introduces a new variable x_n which varies form [0, $(C_i^r.(M-1)^2 + (M-1)^2 + (M-1))/M$]. Therefore, as we can see without making right guesses about the following variables, the adversary will not be able to solve these equations.

2) Known plaintext attacks: In these attacks, the attacker knows the plaintext as well as the corresponding ciphertext. Assuming that the attacker can attain the ciphertext (*C*) and the corresponding plaintext (*P*) of one node she can compute only the session encryption key of that node. However, since the session key expires after usage, it will do her no good. If the adversary attains ciphertext and plaintext of *N* nodes at session *s*, she can form the below system of equations:

C_1^s= $P_1^s \oplus K_1 \oplus co_1.\Delta^s$
....
C_N^s= $P_N^s \oplus K_N \oplus co_N.\Delta^s$

As seen, this system has *N* equations with 2*N*+1 variables, therefore the adversary can neither compute the secret keys nor the addition value.

Moreover, if an adversary has the plaintext and corresponding ciphertext of more than one node in different sessions she will be able to form the following system of equations:

C_1^{s1}= $P_1^{s1} \oplus K_1 \oplus co_1.\Delta^{s1}$ C_1^{sr}= $P_1^{sr} \oplus K_1 \oplus co_1.\Delta^{sr}$
….. …..
C_N^{s1}= $P_N^{s1} \oplus K_N \oplus co_N.\Delta^{s1}$ C_N^{sr}= $P_N^{sr} \oplus K_N \oplus co_N.\Delta^{sr}$

As we can see by subtracting the equations of each line the adversary will achieve:

α_1= $co_1.(\Delta^{s2} - \Delta^{s1})$
....
α_N= $co_N.(\Delta^{s2} - \Delta^{s1})$

These equations are different coefficients of $(\Delta^{s2} - \Delta^{s1})$ and by having all of them the adversary has only one effective equation. By eavesdropping another session, one effective equation and another variable (Δ) is achieved. Therefore, the adversary will not be able to compute neither of the unknowns.

3) Blind rejection: Our main aim in this article is to prevent any external attacks to corrupt the final aggregation result and cause blind rejection. Our protocol overcomes blind rejection by verifying encrypted messages and stops any invalid data from reaching the sink.

We try to illustrate our protocol's reaction to the presence of external attacks. In CMT, presence of a single external attack leads to 100% data loss. However, in our scheme packet loss depends on the level at which the attack has been performed. When the attack is close to the leaf nodes, the packet loss is less. As the attack gets closer to the base the loss becomes more sensible. In an N-ary tree of depth d the packet loss, due to an attack at *x*th level, can be computed by the equation:

$$Pl = \frac{\sum_{k=1}^{d-x+1} N^k}{\sum_{k=0}^{d} N^k} \tag{27}$$

B. Energy cost

1) Computation cost: without concerning the cost of updating session keys, compared to CMT in our method two additional computational concepts have been used: middle way validation and dummy values. It might seem that these impose further computational cost. However, this is only true in the absence of silent nodes, where the message size in CMT is fixed. While at presence of silent nodes (sensors and aggregators), in CMT each aggregator needs to process and construct larger packets and therefore, consume more CPU energy. Depending on the number of silent sensors, this energy consumption might exceed that of our method. This is shown in our simulation results. Furthermore, in CMT each sensor node at each session computes 3 pseudorandom functions and 3 hash functions while in our method one pseudorandom function is computed.

2) Transmission overhead: In this part we compare this scheme to other schemes in transmission cost. We focus on typical works which provide confidentiality and aggregate integrity. For this purpose we compare the existing work with CMT method and combination of a hop-by-hop scheme with a witness based [5] approach. Assume that |*Sensors*| denotes the number of sensors in the network and |*data*| denotes the bit size of data sensed by each sensor node. Therefore, the aggregation of all sensors' ciphertexts or checksums will result in a $log_2|Sensors|$ + $|data|$ bits field.

In CMT method for sending a ciphertext and a checksum (2.($log_2|Sensors|$ + |data|)) bits are needed. For the base station to be able to decrypt the aggregated message it is needed that at least the id of non-contributing sensor nodes be appended to the message. Therefore, if M_a is the number of non-contributing nodes from aggregator a then $M_a. log_2|Sensors|$ bits will be sent along the ciphertext and its checksum.

The message in our method consist of a ciphertext, a checksum, and a dummy counter. This dummy counter will represent the number of silent sensor nodes which can be at most the number of all the sensor nodes in the aggregation tree. Therefore, the size of this counter is always $log_2|Sensors|$ bits.

In a simple hop-by-hop scheme the aggregator receives any incoming message, decrypts it and perform aggregation on decrypted messages and forward the encrypted aggregation result without appending any ids to the message. This scheme only provides confidentiality. To be able to compare this scheme with the other two. We use its combination with the witness based approach which provides aggregate integrity. In this approach, K witness nodes should provide proof for each aggregator. The witness nodes will perform the aggregation and send its MAC to the aggregator. Each aggregator needs to send the encrypted aggregation result, the MAC of aggregation result and K number of MACs provided by its witness nodes. Hence, each message transferred by aggregators contains $log_2|Sensors| + |data|$ +(K+1).$|MAC|$ bits.

If tc is the transmission cost per bit, then you can compare the transmission cost of each aggregator of current method, CMT, and a witness base hop-by-hop scheme by table 1.

TABLE I. TRANSMISSION COSTS COMPARISON

Transmission Costs		
Current method	**CMT**	**Hop-by-Hop Witness based approach**
$tc(2(log_2|sensors|+|data|) + log_2|Sensors|)$	$tc(2(log_2|sensors|+|data|) + M_a.log_2|Sensors|)$	$tc(log_2|sensors| + |data| +(K+1).|MAC|)$

In this part, we compare the three methods in terms of number of bits sent per node in a 3-ary tree. Considering that the number of sensor nodes (leave nodes) are 6561 and the possible reported value by sensors is a number between 1 and 40, the proper choice for the value M would be 40×6561 = 262440, and the encrypted message and their checksums can be represented by a 19 bit fields. The dummy counter can be shown by a 13 bits field. In CMT method, a 13 bits (bits enough for representing all tree nodes) field will be appended to the message for each idle sensor node. For the hop-by-hop witness based approach we assume that we have 4 witnesses for each aggregator and 7 bits MACs. Although the witness based approach is introduced for cluster based aggregation, we assume that we are using it in form of hierarchical tree where each middle way aggregator receives the witness MACs and verifies each child nodes reported value. We base our evaluation on packet formats of TinyOs[20], in which the packet header is 56 bits and the maximum supported data payload of each message is 232 bits. Data payloads over this number will be sent in more than one message. Table 2 shows the number of bits sent per node at different levels of a 3ary tree. The table compares the number of bits sent per node in our method, the witness based hop-by-hop method and the CMT method assuming that 0%, 10% and 30% of leaf nodes remain idle.

TABLE II. NUMBER OF BITS SENT PER NODE IN EACH LEVEL

Level	Number of nodes	Our Method	Witness based hop by hop	CMT (0%)	CMT (10%)	CMT (30%)
1	3	106	110	94	1265	3609
2	9	106	110	94	465	1265
3	27	106	110	94	199	464
4	81	106	110	94	129	198
5	243	106	110	94	106	129
6	729	106	110	94	98	105
7	2187	94	82	94	84	66

In our method, the message in each layer consists of a ciphertext, a checksum, and a dummy counter. Only the leaf nodes do not need to send a dummy counter. By increasing the number of leaf silent nodes, only the average number of bits in leaf nodes decreases and the aggregators will still send a constant number of bits. In a witness based hop-by-hop method each middle way aggregator will send an encrypted message and 5 MACs to the upper level aggregator. By increasing the silent leaf nodes no additional Ids need to be sent. Alternatively, in CMT method only if all nodes reply, a constant number of bits are sent by each aggregator, which is smaller than that of our method and the hop-by-hop method except for the final level. However, in CMT method increasing the number of silent leaf nodes (10% and 30%) increases the number of bits sent per node as we get closer to the base. The reason is that the Ids of noncontributing nodes are appended to the message and as mentioned above, when the number of appending Ids increases, sometimes more messages should be sent for transmitting one aggregation result.

To better understand the transmission and computational costs of our method compared to CMT, we implemented our protocol and CMT protocol with tinyos-2.1.0 and evaluated energy cost in each node with Avrora simulator. The following graphs show the (CPU + radio) energy costs spent at each aggregator node, upon updating the session keys and dummy values, sensing , receiving a message from successor nodes, aggregating and forwarding the result to the predecessor, without considering the cost of computing the pseudorandom function, when there exists no silent nodes in the lowest levels of a 2, 3, and 4ary trees and when a fraction of leaf successors of aggregators are silent at each layer (1/2 in 2-ary, 1/3 in 3-ary and 1/4 in 4-ary). We assume that in CMT method the aggregators perform the sensing as well as aggregating. By comparing the results we can infer that when there is not silent node in the network the CMT method outperforms our scheme in transmission cost because in our method a dummy counter is always appended to the message. It also outperforms in computational cost since in our method additional verification and updating dummy values is performed. However, our scheme performs better at presence of silent nodes, especially at layers closer to the base.

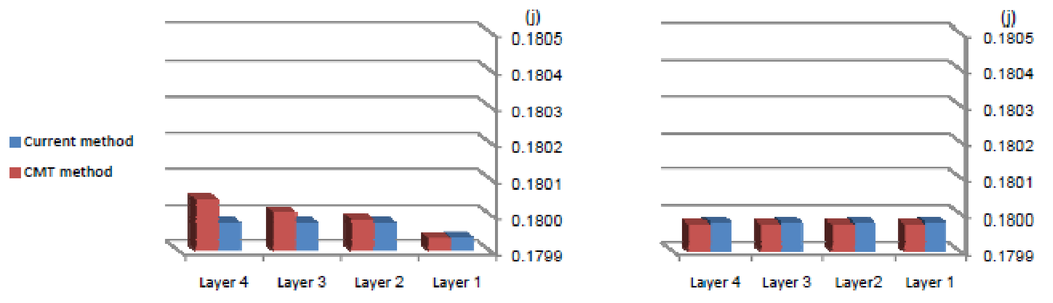

Figure 1. The graph on the left shows CPU and radio energy consumption at presence of silent nodes in different layers of a 2-ary tree and the graph on right shows this amount at absence of silent nodes.

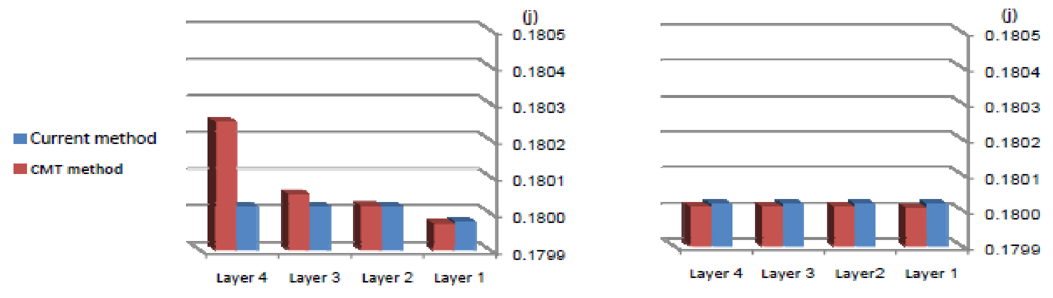

Figure 2. The graph on the left shows CPU and radio energy consumption at presence of silent nodes in different layers of a 3-ary tree and the graph on right shows this amount at absence of silent nodes.

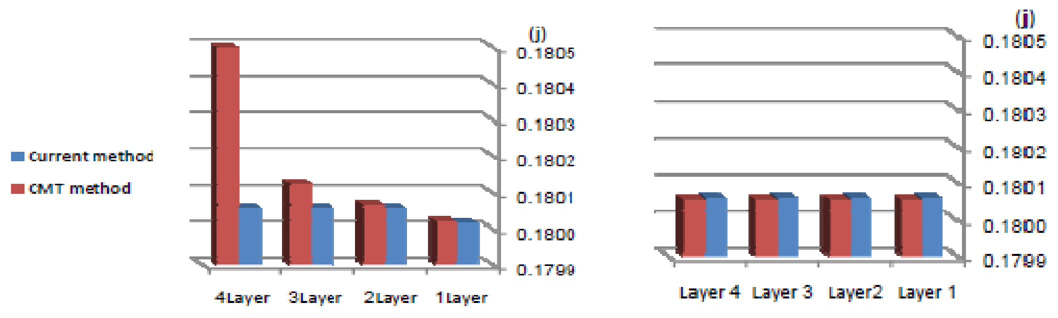

Figure 3. The graph on the left shows CPU and radio energy consumption at presence of silent nodes in different layers of a 4-ary tree and the graph on right shows this amount at absence of silent nodes.

Conclusion and future direction

In this paper we have proposed a CDA derivative for providing end-to-end confidentiality and aggregate integrity. We have proposed aggregate checksums which can be used to provide aggregate integrity through a distributed validation scheme. Each data aggregator can verify its children's aggregated value and checksum, and drop the invalid messages before corrupting correct measurements. This validation scheme avoids referring to the Base Station and blind rejection of valid data. In this scheme also, updateable dummy values and checksums are used to provide robustness at presence of silent sensors. All encryption keys and values are updated at each session to provide robustness against cryptographic attacks.

Our future work entails further enhancing this scheme through a dynamic structure for expelling attacked nodes from the aggregation tree and rearrangements.

REFERENCES

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A survey on sensor networks,” IEEE Communications Magazine. vol. 40, no. 8, pp. 102– 114, November 2002.

[2] C. Karlof, D. Wagner, “Secure routing in wireless sensor networks: attacks and countermeasures,” Elsevier’s AdHoc Networks Journal, vol. 1, no. 2–3, pp. 293–315, 2003.

[3] L. Hu, and D. Evans, “Secure aggregation for wireless networks,” In Proc. Workshop on Security and Assurance in Ad Hoc Networks, pp. 384-391, January 2003.

[4] B. Przydatek, D. Song, and A. Perrig, “SIA: secure information aggregation in sensor networks,” In Proc. SenSys’03, pp. 255–265, July 2003.

[5] W. Du, J. Deng, Y.S. Han, and P.K. Varshney, “A witness-based approach for data fusion assurance in wireless sensor networks,” In Proc. IEEE Global Telecommunications Conference (GLOBECOM ’03), pp. 1435–1439, December 2003.

[6] M. Manulis, and J. Schwenk, “Security model and framework for information aggregation in sensor networks,” ACM Transactions on Sensor Networks, vol 5, no 2, March 2009.

[7] C. Castelluccia,A. Chan,E. Mykletun, and G. Tsudik, “Efficient and provably secure aggregation of encrypted data in wireless sensor Networks,” ACM Transactions on Sensor Networks, vol. 5, no. 3, May 2009.

[8] M. Esler, J. Hightower, T. E. Anderson, G. Borriello. “Next century challenges: data-centric networking for invisible computing,” In Proc. ACM MOBICOM, pp. 256–262, 1999.

[9] C. Intanagonwiwat, D. Estrin, R. Govindan, J. Heidemann. “Impact of network density on data aggregation in wireless sensor networks,” In Proc. IEEE ICDCS, pp. 457- 458, 2002.

[10] Ralph C. Merkle. “Protocols for public key cryptosystems,” In Proc. The IEEE Symposium on Research in Security and Privacy, pp 122–134, April 1980.

[11] Y. Yang, X. Wang, S. Zhu, and G. Cao, “SDAP: a secure hop-by-hop data aggregation protocol for sensor networks,” ACM Transactions on Information and Systems Security, vol. 11, no. 4, July 2008.

[12] S. Ozdemir , Y. Xiao , “Secure data aggregation in wireless sensor networks: A comprehensive overview,” Elsevier Computer Networks, vol. 53, no. 12, pp. 2022–2037, 2009.

[13] J. Girao, D. Westhoff, and M. Schneider, “CDA: Concealed data aggregation in wireless sensor networks,” In Proc. ACM Workshop on Wireless Security (WiSe ’04), October 2004.

[14] C. Castelluccia, E. Mykletun, and G. Tsudik, “Efficient aggregation of encrypted data in wireless sensor networks,” In Proc. Conference on Mobile and Ubiquitous Systems: Networking and Services, pp. 109–117, July 2005.

[15] E. Mykletun, J. Girao, and D. Westhoff, “Public key based cryptoschemes for data concealment in wireless sensor networks,” IEEE Internationam Conference on Communications (ICC'06), pp. 2288-2295, June 2006.

[16] J. Domingo-Ferrer, “A provably secure additive and multiplicative privacy homomorphism,” In Proc. Information Security Conference, pp. 471–483, 2002.

[17] S. Huang, S. Shieh, and JD. Tygar “Secure encrypted-data aggregation for wireless sensor networks,” Springer Wireless Networks, vol. 16, no. 4, pp. 914- 927, 2010.

[18] F. Armknecht, D.Westhoff, J. Girao, and A. Hessler. “A lifetime-optimized end-to-end encryption scheme for sensor networks allowing in-network processing,” Elsevier Computer Communications, vol. 31, no. 4, pp. 734–749, 2008.

[19] O. Goldreich, S. Goldwasser, and S. Micali, “How to construct random functions, ” Journal of the ACM, vol. 33,no. 4, pp. 792-807, October 1984.

[20] C. Karlof, N. Sastry, D. Wagner, “Tinysec: a link layer security architecture for wireless sensor networks,” In Proc. The ACM Conference on Embedded Networked Sensor Systems (SenSys), pp. 162–175, 2004.