2011 International Symposium on Computer Networks and Distributed Systems (CNDS), February 23-24, 2011

A Distributed Verification Scheme for Encrypted Data Aggregation 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 composed of sensors with low computational and energy resources. Transmission of data is one of the most energy consuming operations in such networks. In-network data aggregation is a popular technique which is performed to reduce data transmissions. However, by aggregating multiple data into one, the security of them is no longer guaranteed. While concealed data aggregation methods have been recently proposed to provide energy efficient, end-to-end confidentiality, not much work has been done to enhance them with aggregate integrity. In this work, we propose a scalable and energy efficient bihomomorphic method to preserve end-to-end confidentiality and aggregate integrity against outsider attacks. A distributed validation scheme is used to prevent blind rejection caused by outsiders. To provide better resilience, a key refreshment method is used to prevent analysis attacks.**

Keywords-Concealed data aggregation; wireless sensor networks; security; confidentiality; aggregate integrity

I. INTRODUCTION

Wireless sensor networks are becoming increasingly popular due to their vast potential domain of applications, such as military settings, law enforcements, and environment monitoring. In these networks, nodes collaborate to gather measurements from a monitored area and route them to a processing center. Due to the limited power of each node, a major challenge in such networks is energy efficiency, and any protocol should be designed with considering solutions for saving energy [1]. Data aggregation and in-network processing are commonly used as a solution to the named problem. By combining and summarizing data from different sensor nodes, data aggregation reduces the amount of data transmissions. Data aggregation can be performed by using any algebraic function or statistical operation with multiple inputs and one output, such as addition, multiplication, 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, and providing the security of aggregated data in such networks is of utmost importance. Accordingly, designing aggregation methods with the essential security requirements, such as end-to-end confidentiality, integrity, authentication, and availability has become an active field of research.

Many solutions have been proposed to provide security for aggregated data. End-to-end confidentiality is provided through a concept called concealed data aggregation, which ensures that plaintext of data is not disclosed to any authorized or unauthorized member. However, it is impossible to provide end-to-end integrity as the aggregation function itself changes the original data. Alternatively, there have been some constructions to provide aggregate integrity [3-6]. Aggregate integrity ensures that the aggregated data has not been tampered en-route. In most proposed methods, the aggregated data is verified through a call back from the Base Station or aggregator nodes to the downstream nodes. This type of verification imposes further data transmission on the network. Furthermore, the verification is mainly done in a centralized manner at the Base Station and no verification is performed prior to the arrival of the result. Another drawback of this style of verification is that a single attack on a sensor reading will lead to rejecting all valid readings, which have contributed in the aggregation procedure and have been routed to the Base.

Contributions: This paper proposes a bihomomorphic method for aggregating encrypted data. Our main contribution is to enhance the concealed data aggregation method proposed in [7] with a distributed data validation mechanism. Using this scheme, outsider only attacks are diagnosed and eliminated before corrupting valid aggregated message. This method can provide fairness in terms of forwarding cost per node, and all necessary parameters are updated at the end of each session to provide robustness.

Organization: The rest of the paper is organized as follows. The related works are reviewed in section II. Network and attacker model are described in section III. We introduce our protocol and algorithmic details in sections IV and V. The overall security evaluation and performance analysis are presented in section VI. Section VII presents the conclusion and future directions.

II. Related Works

Early data aggregation schemes did not consider security [8, 9]. Hu and Evans [3] proposed a method for detecting node misbehaviors through delayed aggregation and delayed authentication. This protocol ensured integrity but did not address confidentiality. In [4], the correctness of aggregation result is checked through random sampling and interactive proofs, and a Merkle Hash Tree [10]. In [5], the correctness of data is assured through collecting information from witness nodes. Witnesses are the nodes that perform data aggregation and compute MACs (Message authentication code) of aggregated data but only send the MACs to the Base Station. These MACs are used for the verification purpose at the Base Station. A secure hop-by-hop method, based on divide-and-conquer and commit-and-attest is proposed in [11]. In this method, the trust on high level aggregators is reduced to alleviate the potential security risk caused by a compromised high level node.

By combining traditional cryptography algorithms and data aggregation in above-mentioned methods, messages need to be encrypted hop-by-hop. It means that in order to perform data aggregation, intermediate nodes need to decrypt the messages, aggregate them, and then encrypt the result before forwarding it. This routine causes energy waste, additional delay, and security risks [12]. In order to achieve end-to-end confidentiality privacy homomorphic cryptography has been used [7, and 13-15]. Privacy homomorphism is a transformation which performs direct computations on encrypted data. An additive homomorphic method over data set Q is formulized as follows, where K_{en} and K_{de} are the encryption and decryption keys while D and E denote decryption and encryption:

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

Concealed data aggregation (CDA), was first introduced in [13], where an additive/multiplicative privacy homomorphism method [16] was used for aggregating ciphertext. Although this method provides 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 method using modular addition is introduced in [14]. This method is further extended in [7] by implementing an aggregate authentication scheme, which can recognize outsider only attacks at the Base Station. Lately, an aggregation method on encrypted data is proposed in [17], which can compare messages and eliminate redundant readings without decrypting them. Moreover, the comparison between public key based privacy homomorphism methods in [15] has shown that for the networks where energy is the most important factor, application of symmetric key methods is the best option.

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 $(k_1 \dots . k_\alpha) \in \mathbb{K}_\alpha$ and $(v_1 \dots . v_\alpha) \in \mathbb{P}^\alpha$, $(\alpha \geq 1)$:

$$D_{K_\alpha(k_1 \dots k_\alpha)}(\oplus_{i=1}^{\alpha} E_{k_i}(v_i)) = \sum_{i=1}^{\alpha} v_i \quad (2)$$

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 $k_1 \dots k_\alpha \in \mathbb{K}$ and the key generator kg there is a key $k \in \mathbb{K}$ such that:

$$kg(k_1, t) \odot \dots \odot kg\,(k_\alpha, t) = kg(k, t) \quad (3)$$

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 $n \geq 1$, $E_k(m) = k + m \bmod n$ is a bihomomorphic encryption.

III. Network and Attacker Model

A. *Network model*

We consider a large, densely deployed sensor network with |*N*| nodes, which is arranged in a topological tree. Each node identifies its parent and child nodes. Leaf nodes sense, encrypt, and forward the result to the aggregators. Aggregators perform aggregation on the encrypted data, and forward the result. There is a single node *BS* as the Base Station with unlimited power, which performs decryption. Each node S_i is a wireless sensor node such as Miza2 motes. Due to session key refreshments, our aggregation method requires that all nodes send their data to the Base Station at the same time intervals. Therefore, we assume that nodes are loosely synchronized with a constant period. The key refreshment requires that each epoch denoted by *t*, consists of *r* intervals. The value of *r* is decided before deployment.

We assume that each node performs only one measurement per interval. In each interval, 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 destination. Sensors and aggregators have preinstalled setup values which are as follows:

1) *Sensor's setup values:* Before deployment, each sensor node S_i, receives a group key KG, *r* number of private encryption keys $K_i^1 \dots K_i^r$, and *r* number of private checksum

keys $K_i^{\prime 1} \ldots K_i^{\prime r}$, which form the master keys $\overline{K^1} \ldots \overline{K^r}$ and $\overline{K'^1} \ldots \overline{K'^r}$ for the next r intervals. These master keys are stored at the base:

$$\overline{K^{m\epsilon\, 1\ldots r}} = \odot_{i=1}^{N} K_i^m \ , \ \overline{K'^{m\epsilon\, 1\ldots r}} = \odot_{i=1}^{N} K_i'^m \qquad (4)$$

2) *Aggregator's setup values*: Each aggregator node *a* will receive the aggregated value of its leaf successors' checksum keys, $K_a'^{m\epsilon\, 1\ldots r} = \odot_{i\, \epsilon\, succ(a)} K_i'^m$, and the group key *KG*. It will also receive encrypted dummy values $D_{i\, \epsilon\, child(a)}^{m\epsilon\, 1\ldots r} = \oplus_{n\, \epsilon\, succ(i)} (R \oplus K_n^m)$ and dummy checksums $Dc_{i\, \epsilon\, child(a)}^{m\epsilon\, 1\ldots r} = KG.\, D_i^m \oplus (\oplus_{n\, \epsilon\, succ(i)} (K_n'^m))$ for each of its child nodes *i*. Dummy values are encryption of a unique value *R* which will replace the unreported value of silent nodes. By *succ(a)* we mean all the sensor offspring of the aggregator *a*. For example, *succ(BS)* denotes all the leaf sensor nodes in the tree. By *child(a)* we mean direct children of *a*.

3) *Key update*: Since the applied encryption method is deterministic, there should be a key refreshment scheme. After each *r* intervals, the private encryption keys, the aggregated checksum keys, and encrypted dummy values can be refreshed for the next *r* interval to make the scheme more resilient against cryptographic attacks, provided that the sink is aware of the aggregation of the refreshed keys [18]. Each key should be updated in a manner that the master keys at the base can be updated:

$$\overline{K^{n\epsilon 1\ldots r}[t]} = \overline{K^n} \odot \overline{\Delta_{BS}[t]} \qquad (5)$$

$$\overline{K'^{n\epsilon 1\ldots r}[t]} = \overline{K'^n} \odot \overline{\Delta_{BS}[t]} \qquad (6)$$

During the key refreshment phase, the base will split $\overline{\Delta_{BS}[t]}$ and send $\overline{\Delta_N[t]}$ to each of its immediate children *N* such that $\overline{\Delta_{BS}[t]} = (\odot_{i\, \epsilon\, child(S)} \overline{\Delta_N[t]})$.

Each intermediate node *N* which has received $\overline{\Delta_N\, [t]}$ from its parent will compute and send checksum update values ($\overline{\Delta_{N'}[t]}$) to its children:

$$\overline{\Delta_N[t]} = (\odot_{N'\epsilon\, child(N)} \overline{\Delta_{N'}[t]}) \qquad (7)$$

This node can also update its own aggregated checksum keys and dummy values by:

$$K_N'^{n\epsilon\, \{1\ldots r\}}[t] = K_N'^n \odot \overline{\Delta_N[t]} \qquad (8)$$

$$D_{N'\, \epsilon\, child(N)}^{n\epsilon\, \{1\ldots r\}}[t] = D_{N'}^n \odot \overline{\Delta_{N'}[t]} \qquad (9)$$

The above procedure is continued until all aggregated checksum values are refreshed and the update values reach leaf nodes. The leaf nodes will update their keys when they receive their update value $\Delta_i[t]$ from their parent:

$$K_i^{n\epsilon\, \{1\ldots r\}}[t] = K_i^{n\epsilon\, \{1\ldots r\}} \odot \Delta_i[t] \qquad (10)$$

$$K_i'^{n\epsilon\, \{1\ldots r\}}[t] = K_i'^{n\epsilon\, \{1\ldots r\}} \odot \Delta_i[t] \qquad (11)$$

Like the original work using the group key *KG* makes the scheme vulnerable to malicious insiders. We assume that the group key is updated with a pseudorandom function (*PRF*) from the master key *KG* at each interval. If $f(\cdot)$ is the *PRF*, *Ntr* is the nonce used for epoch *t* phase *r*, and *KG* is the initial group key, then the group key of epoch *t* interval *r* is computed as:

$$KG^r[t] = (KG.\, f_{KG}(Ntr)) \bmod M \qquad (12)$$

Dummy checksums need to be updated at each interval as well:

$$Dc_{N'\, \epsilon\, child(N)}^r[t] = \left(KG^r[t].\, D_{N'}^r[t] \oplus K_{N'}'^r[t]\right) \qquad (13)$$

Although the aggregators do not have the aggregated checksum key of each child node ($K_{N'}'^r[t]$), they should still be able to update the dummy checksums. For this purpose, by using (8), (9), and (12), we will rewrite (13):

$$((KG.\, f_{KG}(Ntr)) \,.\, (D_{N'}^r \odot \overline{\Delta_{N'}[t]})) \oplus (K_{N'}'^r \odot \overline{\Delta_{N'}[t]}) \qquad (14)$$

If we add and remove 1 to $f_{KG}(Ntr)$ in (14) we will have:

$$(KG.\, (1 - 1 + f_{KG}(Ntr)).(D_{N'}^r \odot \overline{\Delta_{N'}[t]}) \oplus (K_{N'}'^r \odot \overline{\Delta_{N'}[t]}) = \qquad (15)$$

$$(KG.D_{N'}^r \oplus K_{N'}'^r) \oplus (KG.\overline{\Delta_{N'}[t]}) \oplus (KG.\, (f_{KG}(Ntr) - 1).(D_{N'}^r[t])) \oplus \overline{\Delta_{N'}[t]} \qquad (16)$$

We can replace ($KG\, .\, D_{N'}^r \oplus K_{N'}'^r$) with $Dc_{N'}^r$ in (16). Therefore, each intermediate aggregator *N* can update its dummy checksums through:

$$Dc_{N'\, \epsilon\, child(N)}^r[t] = Dc_{N'}^r \oplus ((KG + 1).\, \overline{\Delta_{N'}[t]}) \oplus (KG.\, (f_{KG}(Ntr) - 1) \,.\, (D_{N'}^r[t])) \qquad (17)$$

B. Attacker model

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

- We assume a global attacker, who is capable of eavesdropping communication between nodes without compromising any 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.

IV. Approach

Our work is motivated by the work described in [7], which is called CMT. In CMT method, the encrypted data are aggregated using privacy homomorphism, and aggregate authenticates are used against outsider only attacks. However, the result of false aggregate is only found and rejected at the Base Station and no verification is performed before the arrival of the aggregated data at the Base. Therefore, only a single malicious node can cause blind rejection, leading to destruction of a large amount of correct data. This method is also not scalable in terms of silent nodes, as, the Base Station needs the Id of the contributors for the purpose of decryption. These Ids are added to the message and expand it at each aggregator. In this paper, by computing checksums from ciphertext rather than plaintext, we introduce a method which allows middle way aggregation and verification of the encrypted data.

V. Proposed Model

We will first 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. The message and its checksum are sent to the aggregator node. Each aggregator waits for a short period to receive all measurements of child nodes. If, for any reason, any child's measurement is not received, an encrypted dummy value and checksum will replace its message so that the Base Station would not need its Node Id for the purpose of decryption. Upon receiving the encrypted values and their checksums, the aggregator node will aggregate them. For verification purpose, each aggregator node has the group key, and the sum of private checksum keys used by its children. Through this information, the aggregator can validate the aggregated messages and aggregated checksums. An invalid aggregate checksum notifies the aggregator about 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 and compute an aggregate value and checksum from the encrypted dummy value of each node. This dummy aggregate represents 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 added to the dummy counter. The final result of aggregation, either dummy or real, will be sent to the upper level aggregator together with the dummy counter. These dummy values will be finally removed from the final result at the Base Station.

In the following sections, 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, $a \oplus$ b denotes (a+b mod M) and we assume that M is sufficiently large such that if N different ciphers are added then $\sum_{i=1}^{N} m_i \leq M$. We also assume that each epoch t is divided to r phases.

A. *The algorithm executed by leaf node S_i at epoch t phase r*

1) Compute $KG^r[t]$, $K_i^r[t]$, $K_i'^r$[t];

2) Sense plaintext message $m_i \epsilon$ *[0, M-1] ;*

3) Encrypt message $m_i : c_i = m_i \oplus K_i{}^r[t]$*;*

4) Compute checksum*:* $cs_i = (KG^r[t].c_i) \oplus K_i{}'^r[t]$*;*

5) Send (c_i, cs_i) to the parent node*;*

B. *The algorithm executed by aggregator a at epoch t phase r*

1) Compute $KG^r[t]$, $K_a'^r[t]$, $D^r_{i \, \epsilon \, child(a)}[t]$, $Dc^r_{i \, \epsilon \, child(a]}[t]$*;*

2) Replace a dummy ciphertext and a dummy checksum for each node i the message of which is not arrived: c_i=$D_i^r[t]$, cs_i=$Dc_i^r[t]$, and $ctr_i = |n_i|$ *;*

3) Compute aggregate ciphertext: $c_{agg} = \oplus_{i \, \epsilon \, child(a)} c_i$ *;*

4) Compute aggregate checksum: $cs_{agg} = \oplus_{i \, \epsilon \, child(a)} cs_i$*;*

5) Update dummy counter: *ctr* = $\sum_{i \, \epsilon \, child(a)} ctr_i$;

6) Validate aggregated message:

if *(*$cs_{agg} == KG^r[t].c_{agg} \oplus K_a'^r[t]$ *)* set *v=1,* else set *v = 0;*

7) If *v= 0,* compute the dummy aggregate and checksum of all child nodes: $c_{agg} = \oplus_{i \, \epsilon \, child(a)} D_i^r[t]$, $cs_{agg} = \oplus_{i \, \epsilon \, child(a)} Dc_i^r[t]$, and *ctr*=$|n_a|$*;*

8) Send ((c_{agg}, cs_{agg}, ctr)) to the parent node*;*

VI. Analysis

A. *Security analysis*

1) Blind rejection: In this paper, we try to prevent any external attacker from corrupting the final aggregation result and causing blind rejection at the Base Station. Our protocol overcomes blind rejection by verifying encrypted messages and stops any invalid data from reaching the Base Station.

In CMT method, the presence of a single external attack leads to 100% data loss. However, in our scheme the packet loss depends on the level at which the attack has been performed. If the attack is close to the leaf nodes, the packet loss is less, but 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 as:

$$Pl = \frac{N^{d-x+1}}{N^d} \tag{18}$$

2) False data injection from outsiders: Unless using checksums, any external attacker can add a random value to the encrypted data without being noticed. Even unreliable data transfer lines can change the content of the messages. Computing the aggregatable checksums as mentioned, provides the ability of verifying the integrity of the message from outsider attacks. However, access of the attacker to the group key would be a security breach.

B. *Overhead analysis*

1) Transmission overhead: Compared to CMT method, in our scheme the transmission overhead is greatly reduced in the presence of silent nodes by using dummy values. In our method, each sent message consists of a ciphertext, a checksum, and a counter, while in CMT each message consists of a ciphertext, a checksum, and the list of Ids of contributing or noncontributing nodes. This list expands at higher levels of the tree, in such a way that at higher numbers of contributing nodes, more transmission burden is imposed on the forwarding nodes. We first compare the transmission cost of the message at aggregator a obtained from our method and CMT, as given in Table I. Here, M_a is the number of offspring nodes which have contributed in the aggregation ($M_a \leq N$), tc is the transmission cost per bit, and $|em|$, $|cs|$, $|cnt|$, $|\Delta|$ represent the number of bits in encrypted message, checksum, the counter and the update message respectively, and c is the number of direct children of a. Since an update messages is transferred to each child after each r intervals we multiply its cost by $\frac{c}{r}$ and add it to the message transmission cost to show its effect. It is obvious that for sending a counter which is at most a number less than N, $\log_2 N$ bits are needed, while for a list of M_a Ids ($|\mathrm{Id}| < N$), $M_a . \log_2 N$ bits are needed.

TABLE I. TRANSMISSION COSTS COMPARISON

Transmission Costs	
Current method	***CMT***
$tc.(\lvert em\rvert + \lvert cs\rvert + \log_2 N + \frac{c}{r}\lvert\Delta\rvert)$	$tc.(\lvert em\rvert + \lvert cs\rvert + M_a . \log_2 N)$

Next, we compare the bandwidth requirements of our method with that of CMT through a numerical example. Considering that the number of sensor nodes (leaf node) are 2187 and the highest value that might be reported is 120, the proper choice for the value M would be 120×2187 = 262440. Therefore, the encrypted message and its checksums can be represented by 19 bits fields, and the dummy counter can be shown by a 12 bits field (bits enough for representing leaf nodes). We assume that after 10 intervals a 19 bits update value is sent to each child in a separate packet. In CMT method, a 12 bits field will be appended to the message for each idle sensor node. We base our evaluation on packet format of TinyOs [19], in which the packet header is 56 bits and the maximum payloads of each message are 232 bits. Payloads over this number will be sent in more than one message. Table II. shows the number of bits sent per node at different levels of a 3-ary tree. This table compares the forwarding cost per node of our method and that of CMT method assuming that 10%, 20%, and 30% of sensor nodes remain idle.

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

Number of bits sent per node					
Level	***Number of nodes***	***Our Method***	***CMT (0%)***	***CMT (10%)***	***CMT (30%)***
1	3	129	94	1137	3335
2	9	129	94	442	1137
3	27	129	94	192	442
4	81	129	94	127	192
5	243	129	94	105	127
6	729	129	94	98	105
7	2187	94	94	84	66

In our method, a dummy counter is always appended to the message, and therefore always a constant number of bits are sent by each node at any level (except for last level). Alternatively, in CMT method only if all nodes reply, a constant number of bits are sent by each node, which is smaller than that of our method except for the final level. However, in this method growth in the number of silent leaf nodes (10% and 30%) increases the number of bits sent per node as we get closer to the Base. This is because of the Ids of noncontributing nodes which are appended to the message. As mentioned above, when the number of appended Ids increases, sometimes more than one message should be sent for transmitting one aggregation result.

2) Computational overhead: Compared to CMT, our scheme imposes additional overhead due to checksum validation and computing dummy aggregates. However, this additional overhead prevents further energy loss caused by transmission of invalid data.

VII. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a CDA derivative for providing end-to-end confidentiality and aggregate integrity in wireless sensor networks. Our protocol preserves data integrity against outsider attacks. It also prevents blind rejection of valid aggregated data by performing data validation in a distributed manner. It is shown that this work provides fairness in terms of number of bits sent per node and reduces the transmission costs in the presence of silent nodes. Our future work includes finding methods for expelling attacked nodes from the aggregation tree and its rearrangement, and further analysis of this method through simulation with TinyOs 2.1 and TOSSIM.

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] 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.