Integration of False Data Detection With Data Aggregation and Confidential Transmission in Wireless Sensor Networks

Suat Ozdemir, *Member, IEEE*, and Hasan Çam, *Senior Member, IEEE*

***Abstract*—In wireless sensor networks, compromised sensor nodes can inject false data during both data aggregation and data forwarding. The existing false data detection techniques consider false data injections during data forwarding only and do not allow any change on the data by data aggregation. However, this paper presents a data aggregation and authentication protocol, called DAA, to integrate false data detection with data aggregation and confidentiality. To support data aggregation along with false data detection, the monitoring nodes of every data aggregator also conduct data aggregation and compute the corresponding small-size message authentication codes for data verification at their pairmates. To support confidential data transmission, the sensor nodes between two consecutive data aggregators verify the data integrity on the encrypted data rather than the plain data. Performance analysis shows that DAA detects any false data injected by up to T compromised nodes, and that the detected false data are not forwarded beyond the next data aggregator on the path. Despite that false data detection and data confidentiality increase the communication overhead, simulation results show that DAA can still reduce the amount of transmitted data by up to 60% with the help of data aggregation and early detection of false data.**

***Index Terms*—Data aggregation, data integrity, network-level security, sensor networks.**

I. Introduction

WIRELESS sensor networks are vulnerable to many types of security attacks, including false data injection, data forgery, and eavesdropping [1]. Sensor nodes can be compromised by intruders, and the compromised nodes can distort data integrity by injecting false data. The transmission of false data depletes the constrained battery power and degrades the bandwidth utilization. False data can be injected by compromised sensor nodes in various ways, including data aggregation and relaying. Because data aggregation is essential to reduce data redundancy and/or to improve data accuracy, false data detection is critical to the provision of data integrity and efficient utilization of battery power and bandwidth. In addition to false data detection, data confidentiality is required by many sensor network applications to provide safeguard against eavesdropping.

Manuscript received August 04, 2008; revised August 14, 2009; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor D. Agrawal.

S. Ozdemir is with the Computer Engineering Department, Gazi University, Ankara 06570, Turkey (e-mail: suatozdemir@gazi.edu.tr).

H. Çam is with Altusys Corporation, Trenton, NJ 08648 USA (e-mail: hasan@altusystems.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2009.2032910

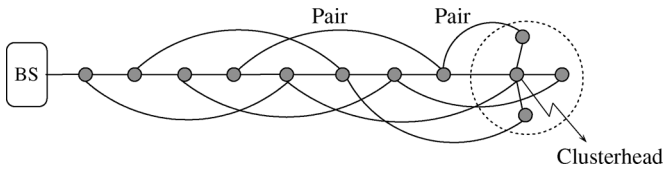

Fig. 1. An example of forming sensor pairs to authenticate data for the false data detection scheme in [3], where data aggregation is not allowed if it requires any change in the data.

To the best of our knowledge, this paper is the first of its kind to integrate the detection of false data with data aggregation and confidentiality.

Data confidentiality prefers data to be encrypted at the source node and decrypted at the destination. However, data aggregation techniques usually require any encrypted sensor data to be decrypted at data aggregators for aggregation. The existing false data detection algorithms [2]–[5] address neither data aggregation nor confidentiality. Although they could be modified easily to support data confidentiality, it is a challenge for them to support the data aggregation that alters data. For instance, the basic idea behind the false data detection algorithm in [3] is to form pairs of sensor nodes such that one pairmate computes a message authentication code (MAC) of forwarded data and the other pairmate later verifies the data using the MAC, as illustrated in Fig. 1. In this scheme, any data change between two pairmates is considered as false data injection, and therefore, data aggregation is not allowed if it requires alterations in the data. Hence, the false data detection algorithm cannot be implemented when a data aggregator between two pairmates changes the data.

Data aggregation is implemented in wireless sensor networks to eliminate data redundancy, reduce data transmission, and improve data accuracy. Data aggregation results in better bandwidth and battery utilization [6], [25], which enhances the network lifetime because communication constitutes 70% of the total energy consumption of the network [7]. Although data aggregation is very useful, it could cause some security problems because a compromised data aggregator may inject false data during data aggregation. When data aggregation is allowed, the false data detection technique should determine correctly whether any data alteration is due to data aggregation or false data injection. A joint data aggregation and false data detection technique has to ensure that data are altered by data aggregation only.

This paper introduces a data aggregation and authentication protocol (DAA) to provide false data detection and secure data aggregation against up to T compromised sensor nodes, for $T \geq 1$. The value of T depends on security requirements, node

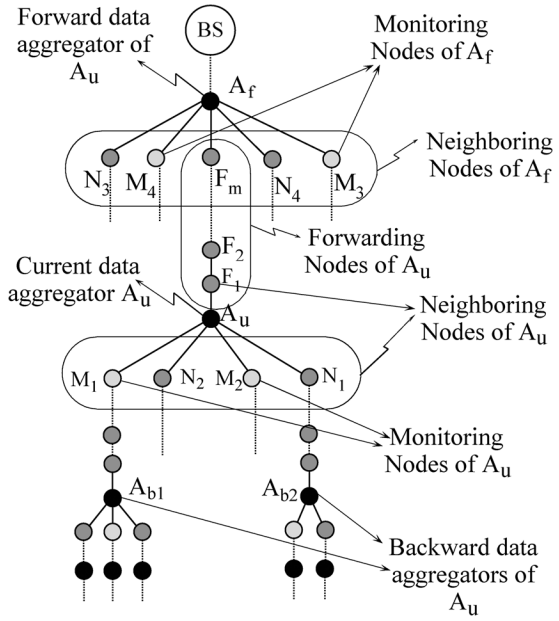

Fig. 2. The system architecture of sensor nodes used by DAA. To support false data detection, secure data aggregation, and confidentiality against up to T compromised sensor nodes, DAA forms $2T + 1$ pairs of sensor nodes by the neighboring and forwarding nodes of A_u and A_f.

density, packet size, and the amount of tolerable overhead. We assume that some sensor nodes are selected dynamically as data aggregators, and the nodes between two consecutive data aggregators are called forwarding nodes simply because they forward data. To detect false data injected by a data aggregator while performing data aggregation, some neighboring nodes of the data aggregator (called monitoring nodes) also perform data aggregation and compute MACs for the aggregated data to enable their pairmates to verify the data later. DAA also provides data confidentiality as data are forwarded between data aggregators. To provide data confidentiality during data forwarding between every two consecutive data aggregators, the aggregated data are encrypted at data aggregators, and false data detection is performed over the encrypted data rather than the plain data. Whenever the verification of encrypted data fails at a forwarding node, the data are dropped immediately to minimize the waste of resources such as bandwidth and battery power due to false data injection. The basic system architecture of DAA is given in Fig. 2.

The remainder of the paper is organized as follows. The related work is presented in Section II. Section III describes the assumptions and limitations of the paper. Protocol DAA is described in Section IV. Section V discusses the performance analysis, and the simulation results are presented in Section VI. Concluding remarks are made in Section VII.

II. Related Work

To detect false data injections, data authentication schemes that employ multiple MACs are presented in [2]–[5]. The statistical en-route detection scheme [2], called SEF, enables relaying nodes and base station to detect false data with a certain probability. In 10 hops, SEF is able to drop 80%–90% of the injected false reports. In the interleaved hop-by-hop authentication scheme [3], any packet containing false data injected by T compromised sensor nodes is detected by those $T + 1$ sensor nodes that collaborate to verify data integrity. In the interleaved hop-by-hop authentication scheme, sensor nodes are not allowed to perform data aggregation during data forwarding. The Commutative Cipher based En-route Filtering scheme (CCEF) [4] drops false data en-route without symmetric key sharing. In CCEF, the source node establishes a secret association with base station on a per-session basis, while the intermediate forwarding nodes are equipped with a witness key. With the use of a commutative cipher [8], a forwarding node can use the witness key to verify the authenticity of the reports without knowing the original session key. In the dynamic en-route filtering scheme [5], false data are filtered in a probabilistic nature in the sense that a forwarding node can validate the authenticity of a report only if it has a corresponding authentication key. A legitimate report is endorsed by multiple sensor nodes using their distinct authentication keys from one-way hash chains.

Secure data aggregation problem is studied extensively [9]–[12] and [28]. In [9], the security mechanism detects node misbehaviors such as dropping or forging messages and transmitting false data. In [10], random sampling mechanisms and interactive proofs are used to check the correctness of the aggregated data at base station. In [11], sensor nodes first send data aggregators the characteristics of their data to determine which sensor nodes have distinct data, and then those sensor nodes having distinct data send their encrypted data. In [12], the witness nodes of data aggregators also aggregate data and compute MACs to help verify the correctness of the aggregators' data at base station. Because the data validation is performed at base station, the transmission of false data and MACs up to base station affects adversely the utilization of sensor network resources. In [28], sensor nodes use the cryptographic algorithms only when a cheating activity is detected. Topological constraints are introduced to build a secure aggregation tree (SAT) that facilitates the monitoring of data aggregators. In SAT, any child node is able to listen to the incoming data of its parent node. When the aggregated data of a data aggregator are questionable, a weighted voting scheme is employed to decide whether the data aggregator is properly behaving or is cheating.

Several key establishment protocols are developed for sensor networks [16], [17], which offer *"direct key establishment"* for neighboring nodes and *"path key establishment"* for sensor nodes that are multiple hops away from each other. In path key establishment method, to establish a pairwise key with node j, a sensor node i needs to find a path between itself and node j such that any two adjacent nodes in that path can establish a pairwise key directly. For example, in [17], to discover a key path to another sensor node, a sensor node picks a set of intermediate nodes with which it has established direct keys. The source node sends a request to all these intermediate nodes. If one of the intermediate nodes can establish a direct key with the destination node, a key path is discovered. Otherwise, this process continues with the intermediate nodes forwarding the request until the destination node is reached.

III. Assumptions and Limitations

We consider a large sensor network with densely deployed sensor nodes. Due to dense deployment, sensor nodes have overlapping sensing ranges so that an event may be detected by multiple sensor nodes, thereby necessitating to aggregate the cor-

related sensed data at neighboring sensor nodes. Some sensor nodes are dynamically designated as data aggregators to aggregate data from their neighboring sensor nodes. To balance the energy consumption of sensor nodes, the role of data aggregator is rotated among sensor nodes based on their residual energy levels. Transmission ranges of data aggregators can be adjusted depending on the number of their neighboring nodes. For instance, if a data aggregator needs more neighboring nodes, its transmission range may be increased. Sensor nodes have limited computation and communication capabilities. For example, the Mica2 motes [13] have a 4-MHz, 8-bit Atmel microprocessor and are equipped with an instruction memory of 128 kB and a RAM of 4 kB. All messages are time-stamped, and nonces are used to prevent reply attacks. Intruders can compromise sensor nodes via physical capturing or through the radio communication channel. Although compromised nodes can perform many types of attacks to degrade the network's security and performance, this paper only considers false data injection and eavesdropping attacks. In the rest of this section, the other assumptions of DAA are explained in detail.

A. Network Topology

We assume that data aggregators are chosen in such a way that: 1) there are at least T nodes, called forwarding nodes, on the path between any two consecutive data aggregators; and 2) each data aggregator has at least T neighboring nodes, so they can form T pairs with the forwarding nodes on the path between two consecutive data aggregators. In order to ensure that there are at least T nodes between any two consecutive data aggregators, we assume that the secure data aggregator selection protocol (SANE) [23] is employed as follows. Sensor nodes are scattered over a large area to form small sets of nodes in close proximity from each other. These sets are called sectors, and the sector size S (i.e., the number of nodes in a sector) depends on the value of T. The protocol SANE is first run to select candidate data aggregators. Since sector size S is determined based on the value of T, the number of intermediate nodes between any two consecutive candidate aggregators is expected to be around T. If it happens that there are less than T intermediate nodes between two consecutive candidate aggregators, one of these candidate aggregators drops its candidacy, and then the protocol SANE is run again. This process is repeated until there are at least T intermediate nodes between any two consecutive data aggregators.

B. Generation of MACs

In DAA, only data aggregators are allowed to encrypt and decrypt the aggregated data. The forwarding nodes first verify data integrity using MACs and then relay the data if it is not false. The TinySec data packet structure [14] includes 29-byte payload and a 4-byte MAC. Although a DAA packet contains the same size of payload (i.e., 29 bytes), it has two 4-byte MACs rather than one 4-byte MAC, leading to a 4-byte increase in the packet length. Each of these 4-byte MACs is called a full-size MAC, denoted FMAC. An FMAC consists of $T+1$ small-size MACs, called subMACs, such that one of them is computed by a data aggregator and the remaining T subMACs are computed by its T monitoring nodes. A subMAC is constructed by selecting some bits of a MAC. To select some bits of a MAC, we assume that each sensor node has the same pseudo-random number generator (PRNG) [15] that generates random numbers ranging from 1 to 32. After the pairs are formed and their shared keys are established, the sensor nodes of each pair initiate their PRNGs using their shared key as the seed. In order for a neighboring sensor node N_i of the current data aggregator $\mathrm{A_u}$ to generate a $\mathrm{subMAC}(D)$ for its pairmate F_j of data forwarding node, the sensor node N_i first computes the $\mathrm{MAC}(D)$ of the data D using the key $K_{i,j}$ that it shares with F_j. Then, assuming that S denotes the size of $\mathrm{MAC}(D)$ in bits, N_i pseudo-randomly selects $S/(T+1)$ bits from $\mathrm{MAC}(D)$ and forms the $\mathrm{subMAC}(D)$. To select the bits from $\mathrm{MAC}(D)$, N_i runs its PRNG $S/(T+1)$ times, which results in $S/(T+1)$ random numbers ranging from 1 to 32. Each random number indicates the index of a bit location in $\mathrm{MAC}(D)$, and the bits of those selected locations constitute the $\mathrm{subMAC}(D)$. To verify this subMAC computed by N_i, its pairmate computes the $\mathrm{MAC}(D)$ and runs its PRNG $S/(T+1)$ times to generate its $\mathrm{subMAC}(D)$. If the subMACs of a pair match, the data D are said to be verified by the pairmate of N_i. Note that PRNGs can be out of synchronization due to packet loses. In DAA, PRNG synchronization is achieved using packet sequence numbers that are added to aggregated data packets. In each data aggregation session, the data aggregator increases the current sequence number by one and informs its neighboring nodes about the current packet sequence number so that each monitoring node synchronizes its PRNGs if needed.

C. Pairwise Key Establishment and Sybil Attacks

It is assumed that a monitoring node can establish a pairwise shared key with its pairmate that is multiple hops away. The existing random key distribution protocols such as [16] and [17] allow key establishment using multihop communication. Although there are some other techniques that allow nonneighboring sensor nodes to establish pairwise keys [26], [27], because of the following reason, random key distribution protocols are preferred in this paper. By using random key distribution protocols, monitoring nodes can ensure the identity of their pairmates, thereby preventing Sybil attacks where a compromised node fakes multiple identities to establish pair relations with more than one monitoring node. As explained in [24], a random key distribution protocol may be used against Sybil attacks if each node ID is associated with the keys assigned to that node and a key validation is performed before establishing pairwise keys.

D. Group Key Establishment

Each data aggregator A_u and its neighboring nodes are assumed to establish a group key, called K^u_{group}, using an existing group key establishment scheme [18]. The group key is used for selecting the monitoring nodes of the data aggregator, and protecting data confidentiality while data are transmitted among data aggregator and its neighboring nodes for data verification and aggregation.

E. Limitations

In this subsection, we list the limitations of DAA due to the above assumptions. First of all, the value of T depends strictly on several factors such as geographical area conditions, modes of deployment, transmission range of sensor nodes, power management, and node density of the network. For example, if the terrain on which sensor nodes are being deployed is constrained

TABLE I
SUMMARY OF NOTATIONS

Notation	Explanation
A_u	Current data aggregator.
A_f	Forward data aggregator.
A_b	Backward data aggregator.
BS	Base Station.
N_i	Neighboring node i of A_u or A_f.
F_j	Forwarding node j of A_u.
M_k	Monitoring node k of A_u.
K^u_{group}	Group key of A_u and its neighbors.
$K_{i,j}$	Shared key between sensor nodes i and j.
$E_{K_{ij}}(D)$	Encryption of data D with key K_{ij}.
$MAC_{K_{ij}}(D)$	Message Authentication Code of data D calculated with key K_{ij}.

by geographical structures (e.g., trees, rocks, hills, etc.), then SANE protocol may not ensure that there are T nodes between some data aggregators. Second, the pairwise key establishment among nonneighboring nodes takes more time than that among direct neighboring nodes [16], [17]. Therefore, such key establishment process is more vulnerable to node compromise attacks, and any intermediate node that is compromised quickly after deployment (i.e., before the key establishment process is over) can obtain the secret key of a sensor node pair. However, in order to successfully mount this attack, the attacker must know when and where the sensor network will be deployed so that he/she can compromise a sensor node quickly after the deployment. Therefore, the success probability of such attack is low. Finally, group communication schemes [18] are vulnerable to those attacks where an adversary who compromises a legitimate group member seizes some or all past group keys as well as the current group key and discloses the secrecy of the data. In [19], some practical methods are introduced to make group communication schemes immune against this type of attacks if the compromised nodes are detected.

IV. DATA AGGREGATION AND AUTHENTICATION PROTOCOL (DAA)

This section presents the protocol DAA and its algorithms, namely MNS and SDFC. The notations that are used in DAA are given in Table I, and the definitions of some terms used throughout the paper are presented next.

Definition 1. (Current Data Aggregator, Backward Data Aggregator, Forward Data Aggregator, Forwarding Node): Let $\mathrm{R} = \{A_0, A_1, \ldots, A_n\}$ represent the set of all data aggregators on a path P from a sensor node to base station BS. The aggregator that we currently consider is called the *current data aggregator*, denoted by A_u, for $1 \leq u \leq n-1$. The previous and next data aggregators of A_u are referred to as its *backward data aggregator* A_b and *forward data aggregator* A_f, respectively, where $b = u - 1$ and $f = u + 1$. Sensor nodes that are located between A_u and A_f on the path P are called the *forwarding nodes* of A_u, as shown in Fig. 2.

DAA provides secure data aggregation, data confidentiality, and false data detection by performing data aggregation at data aggregators and their neighboring nodes and verifying the aggregated data during data forwarding between two consecutive data aggregators. As seen from Fig. 3, DAA has three steps that are explained in the following subsections.

Protocol DAA

Input: A wireless sensor network with densely deployed sensor nodes, some of which are designated as data aggregators. For a given value of T, data aggregators are already selected in such a way that (i) there exist at least T nodes between any two data aggregators, and (ii) each data aggregator has at least T neighboring nodes.

Output: Even though the network can have up to T compromised nodes, data are aggregated in data aggregators, data confidentiality is provided and the injected false data are detected and dropped.

Step 1: T neighboring nodes of each data aggregator are randomly selected as monitoring nodes to perform the additional data aggregation and to compute subMACs of the aggregated data.

Step 2: The following $2T+1$ pairs of nodes are formed by enabling the nodes of every pair to share a distinct symmetric key: (1) one pair is formed by the current and forward data aggregators, (2) T pairs are formed by the monitoring nodes of the current data aggregator and the neighboring nodes of the forward data aggregator, and (3) T pairs are formed by the monitoring and forwarding nodes of the current data aggregator. If two nodes want to form a pair but do not have a shared key, then they are assumed to establish a pairwise shared key using an existing key establishment algorithm.

Step 3: Each data aggregator and its selected T monitoring nodes aggregate data and then compute subMACs. The aggregated data are encrypted by the current data aggregator. The data aggregator and its monitoring nodes compute two subMACs: one subMAC for the encrypted aggregated data and another subMAC for the plain aggregated data. The current data aggregator constructs two FMACs from these subMACs and sends the encrypted data and two FMACs to forwarding nodes. The integrity of the encrypted data is verified by forwarding pairmates of the selected monitoring nodes of the current data aggregator. The integrity of the plain data is verified by some neighboring nodes of the forward data aggregator. If the integrity verification of the encrypted or plain data fails at any sensor node, the data are dropped immediately.

Fig. 3. The protocol DAA.

A. Selection of Monitoring Nodes for an Aggregator (Step 1 of DAA)

In order to perform secure data aggregation, each data aggregator is monitored by its T neighboring nodes out of total n neighboring nodes, for $n \geq T$. Therefore, in the first step of DAA, T neighbors of a data aggregator A_u are selected as *monitoring* nodes to perform the data aggregation and to compute subMACs of the aggregated data. The monitoring nodes are selected by the Monitoring Node Selection (MNS) algorithm, as shown in Fig. 4. The basic idea behind the selection of T monitoring nodes for each data aggregator in Algorithm MNS is to assign indices to the neighboring nodes in some order and then compute T indices by applying modulus operation to the sum of some random numbers generated by the neighboring nodes. Any neighboring node whose index is equal to one of these T indices becomes a monitoring node. The data aggregator and all neighboring nodes are involved with the selection of monitoring nodes to minimize the adverse impact of a compromised node.

Algorithm MNS (Monitoring Node Selection)

Input: Aggregator A_u, its n neighboring nodes, the group key K^u_{group}.
Output: T neighboring nodes of A_u are selected as monitoring nodes.

1: A_u requests its every neighboring node to send 2 random numbers along with its node ID number.
2: Each neighboring node of A_u generates 2 random numbers (R_a and R_b) using its PRNG and the key it shares with A_u. R_a,R_b and MAC($R_a|R_b$) are sent to A_u.
3: When A_u finishes receiving random numbers and node IDs from n its neighboring nodes, it labels them N_i in the receiving order of their random numbers for $1 \leq i \leq n$, so that the nodes that send the random numbers first and last are labeled N_1 and N_n, respectively.
4: A_u sorts all $2 \times n$ random numbers in an ascending order: $R_1, R_2, \cdots, R_{2\times n}$ and computes $MAC(R_1|R_2|\cdots|R_{2\times n})$ using K^u_{group}. Then, A_u broadcasts the sorted random numbers and $MAC_{K^u_{group}}(R_1|R_2|\cdots|R_{2\times n})$ along with node IDs and their new labels.
5: Each N_i verifies the broadcast numbers by checking whether two random numbers R_a and R_b that it sent earlier to A_u match two of the random numbers that A_u has broadcasted. If the verification is successful N_i encrypts the $MAC_{K^u_{group}}(R_1|R_2|\cdots|R_{2\times n})$ using the key it shares with A_u and sends it to A_u. If the verification is not successful, N_i informs its neighboring nodes and A_u about it, along with a request of re-starting the monitoring node selection.
6: To determine the indices of the T monitoring nodes, each N_i runs the following modulus function I_k for $1 \leq k \leq T$. Any N_i whose index i happens to be equal to an I_k is selected as a monitoring node. If there is a duplicate I_k value, modulus function is run again by increasing the k value by 1.

$$I_k = \left[\left(\sum_{j=k}^{n-1+k} R_j + K^u_{group} \right) mod(n) \right] + 1$$

Fig. 4. Algorithm MNS.

As seen from Fig. 4, in Steps 1 and 2, the data aggregator A_u makes a request for random numbers and neighboring nodes to send their random numbers to A_u in the order of their TDMA schedule. Each neighboring node generates the random numbers using its PRNG and the key it shares with A_u. Hence, all neighboring nodes are expected to generate distinct random numbers. In Step 3, A_u assigns indices to the n neighboring nodes according to the order that they send their random numbers. A_u sorts the random numbers in Step 4 in order to enable every neighboring node to compute the same T indices in Step 6. In Step 5, A_u performs broadcast authentication by checking if all neighboring nodes received the random numbers correctly. The formula I_k in Step 6 computes each index by summing up the n random numbers and the group key K^u_{group} such that the n random numbers are picked up by a sliding window of size n. Note that the random numbers are transmitted in plain text and, therefore, the group key K^u_{group} is used in the formula I_k to prevent compromised nodes from knowing the resultant T indices. We assume that an existing group key establishment algorithm [18] is used to establish the group key K^u_{group} for secure communication between A_u and its neighboring nodes.

MNS protects a compromised data aggregator from affecting the monitoring node selection. The monitoring nodes are selected by all neighboring nodes. To affect the selected monitoring nodes, a compromised data aggregator must change the random numbers before broadcasting them. However,

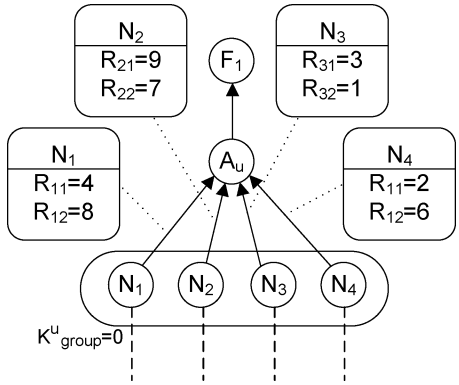

Fig. 5. An example scenario for selection of the monitoring nodes in Algorithm MNS.

in this case the neighboring node whose random number is changed by the data aggregator detects the change. Because monitoring nodes are selected via the random numbers sent by all neighboring nodes, a compromised neighboring node cannot affect the selection process either. Moreover, MNS minimizes the number of transmitted random numbers. Since the maximum value of T is n, at most $n-1+n$ (or approximately $2n$) numbers should be sorted. Therefore, every neighboring node is requested to send two random numbers, resulting in the transmission of $4n$ random numbers. To show how MNS works, an example is given next.

Example 1: Consider the scenario given in Fig. 5. Let us assume that $K^u_{\text{group}} = 0$, $T = 2$ and $n = 4$. Neighboring node N_1 sends random numbers $R_{1_1} = 4$ and $R_{1_2} = 8$, N_2 sends $R_{2_1} = 9$ and $R_{2_2} = 7$, N_3 sends $R_{3_1} = 3$ and $R_{3_2} = 1$, and finally N_4 sends $R_{4_1} = 2$ and $R_{4_2} = 6$ to aggregator A_u. A_u broadcasts the indices and random numbers of these nodes. To determine the monitoring nodes, A_u sorts the random numbers as follows $R_1 = 1$, $R_2 = 2$, $R_3 = 3$, $R_4 = 4$, $R_5 = 6$, $R_6 = 7$, $R_7 = 8$, $R_8 = 9$ and broadcast the sorted random numbers. Since T is 2, two monitoring nodes are selected. Each node runs the modulus function twice and generates the index numbers of the monitoring nodes, $I_1 = [(1+2+3+4+0)\text{mod}(4)]+1 = 3$ and $I_2 = [(2+3+4+6+0)\text{mod}(4)] + 1 = 4$. Therefore, those neighboring nodes that are assigned indices of 3 and 4 are selected as the monitoring nodes of the data aggregator.

B. Forming Pairs of Sensor Nodes (Step 2 of DAA)

DAA assumes that a path already exists between any two consecutive data aggregators via forwarding nodes, and that each data aggregator uses only one outgoing path towards base station at a given time. To establish pairs among monitoring nodes and forwarding nodes, A_f sends out a "pairmate discovery message" M to A_u along with its neighboring node list. A_f also adds the MAC of neighboring node list using the key it shares with A_u. Message M is forwarded by the nodes on the path between A_f and A_u, and each node that forwards M appends its ID to M. When A_u receives M, it has the IDs of its forwarding nodes and neighboring nodes of A_f. Let's assume that there are k forwarding nodes ($k \geq T$) between A_u and A_f. To form the pairs among A_u's monitoring nodes and forwarding nodes, A_u concatenates the IDs of the forwarding nodes in a random order and indexes them 1 to k. Then, A_u computes the MAC of the concatenated IDs using K^u_{group} and broadcasts the MAC

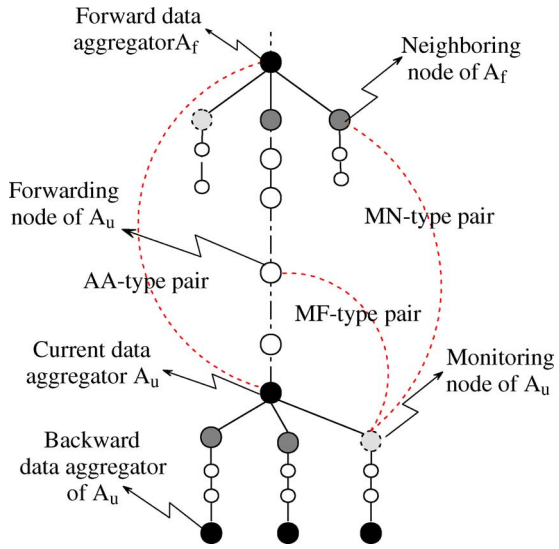

Fig. 6. Node pairs between two consecutive data aggregators A_u and A_f, where $T = 1$. Three types of node pairs are formed: 1) an AA-*type* pair by data aggregators A_u and A_f; 2) an MF-*type* pair by the monitoring node of A_u and a forwarding node of A_u; and 3) an MN-*type* pair by the monitoring node of A_u and a neighboring node of A_f.

and k. Each monitoring node selects an index number between 1 and k, then A_u broadcasts the concatenated ID list so that each monitoring node finds out its pairmate forwarding node. Monitoring nodes verify the correctness of the broadcasted IDs and their indexed orders using the previously broadcasted MAC of the concatenated ID list. Therefore, even if A_u is compromised, it cannot affect the pairmate selection process. The pairs among monitoring nodes and neighboring nodes of A_u are established in a similar fashion.

Two nodes form a pair if they can establish and share a symmetric key for false data detection and data confidentiality. Recall that DAA does not deal with key establishment and assumes that an existing pairwise key establishment algorithm [16], [17] is used to establish a symmetric key between two nodes. As illustrated in Fig. 6, the current data aggregator A_u and the forward data aggregator A_f form one pair of AA-type. If A_u and A_f do not have a shared key, they establish a symmetric key $K_{u,f}$. To form pairs of MF-type by the monitoring and forwarding nodes of A_u, they first send their ID numbers to each other, and then each monitoring node selects a distinct forwarding node as its pairmate. If a monitoring node happens to select a pairmate of forwarding node for which it does not share a key, then they employ a pairwise key establishment algorithm to establish a shared key. Similarly, MN-type of pairs are formed by the monitoring nodes of A_u and the neighboring nodes of A_f.

In Step 2 of DAA, the following $2T + 1$ pairs of nodes are formed: 1) one pair of AA-type is formed by the current data aggregator A_u and the forward data aggregator A_f; 2) T pairs of MF-type are formed by the monitoring and forwarding nodes of A_u; and 3) T pairs of MN-type are formed by the monitoring nodes of A_u and the neighboring nodes of A_f. When T neighboring nodes of A_u are paired up with neighbors of A_f, they sent the IDs of their pairmates to A_u so that A_u can be sure that T neighbors of A_u are paired with T unique neighbors of A_f. Note that if false data detection were not required during data forwarding, then DAA would form only $T + 1$ pairs (i.e., one pair of AA-type and T pairs of MN-type) to detect false data at data aggregators and their neighboring nodes. However, the second step of DAA forms $2T + 1$ pairs to benefit from the fact that false data detection during data forwarding allows false data to be dropped as early as possible.

C. Integration of Secure Data Aggregation and False Data Detection (Step 3 of DAA)

This section introduces Algorithm SDFC to provide false data detection, secure data aggregation and data confidentiality for the third step of DAA. To provide data confidentiality, transmitted data are always encrypted and forwarding nodes perform the data verification over the encrypted data. Prior to this third step of DAA, monitoring nodes of every data aggregator are selected, and $2T + 1$ pairs are formed. To verify data integrity and detect false data injections, one pairmate computes a subMAC, and the other pairmate verifies the subMAC. subMACs are computed for both plain and encrypted data. subMACs of plain data are used to detect false data injections during data aggregation, whereas subMACs of encrypted data are used to detect false data injections during data forwarding. To detect any false data that the current data aggregator A_u can inject *during data aggregation*, the monitoring nodes of A_u also aggregate the incoming data of A_u and compute subMACs for the plain aggregated data, so that the forward data aggregator A_f and its neighboring nodes verify the subMACs. Similarly, to detect those false data that can be injected *during data forwarding*, the monitoring nodes of A_u compute subMACs for the encrypted aggregated data and then their pairmates of forwarding nodes verify the subMACs.

As seen from the flowchart of Algorithm SDFC in Fig. 7, the main steps of SDFC are: 1) whenever some data are received by a data aggregator, the authenticity of data is verified by the data aggregator and its neighboring nodes; 2) the data aggregator and its monitoring nodes aggregate the data independently of each other; 3) each monitoring node computes one subMAC for the encrypted data and the other subMAC for the plain data; 4) the data aggregator collects these subMACs from its monitoring nodes to form the FMACs of the encrypted and plain data, appends the FMACs to the encrypted data, and transmits them; 5) the forwarding nodes verify the data integrity of the encrypted data; and finally 6) the neighboring nodes of the next aggregator verify the integrity of the plain data. In Algorithm SDFC, each data aggregator forms two FMACs: one FMAC for the encrypted data, and the other FMAC for the plain data. Each FMAC consists of $T+1$ subMACs computed by the data aggregator A_u and its T monitoring nodes. The FMACs of encrypted and plain data are forwarded along with the encrypted data in the packet as shown in Fig. 8. In the formation of FMACs, data aggregator A_u determines the order of subMACs in any way and inform each forwarding node about its subMAC location individually. Consequently, an intruder cannot know in advance the exact location of subMAC bits for a given forward node. Therefore, to inject a false message, an attacker has to try all possibilities for a 32-bit FMAC. Thus, if an intruder wants to inject messages at a forwarding node to consume its energy, only $(1/2)^{32}$ of randomly generated messages at a forwarding node can be accepted and forwarded.

Algorithm SDFC is shown in Fig. 9. In Algorithm SDFC, the current data aggregator first collects data from its neighboring

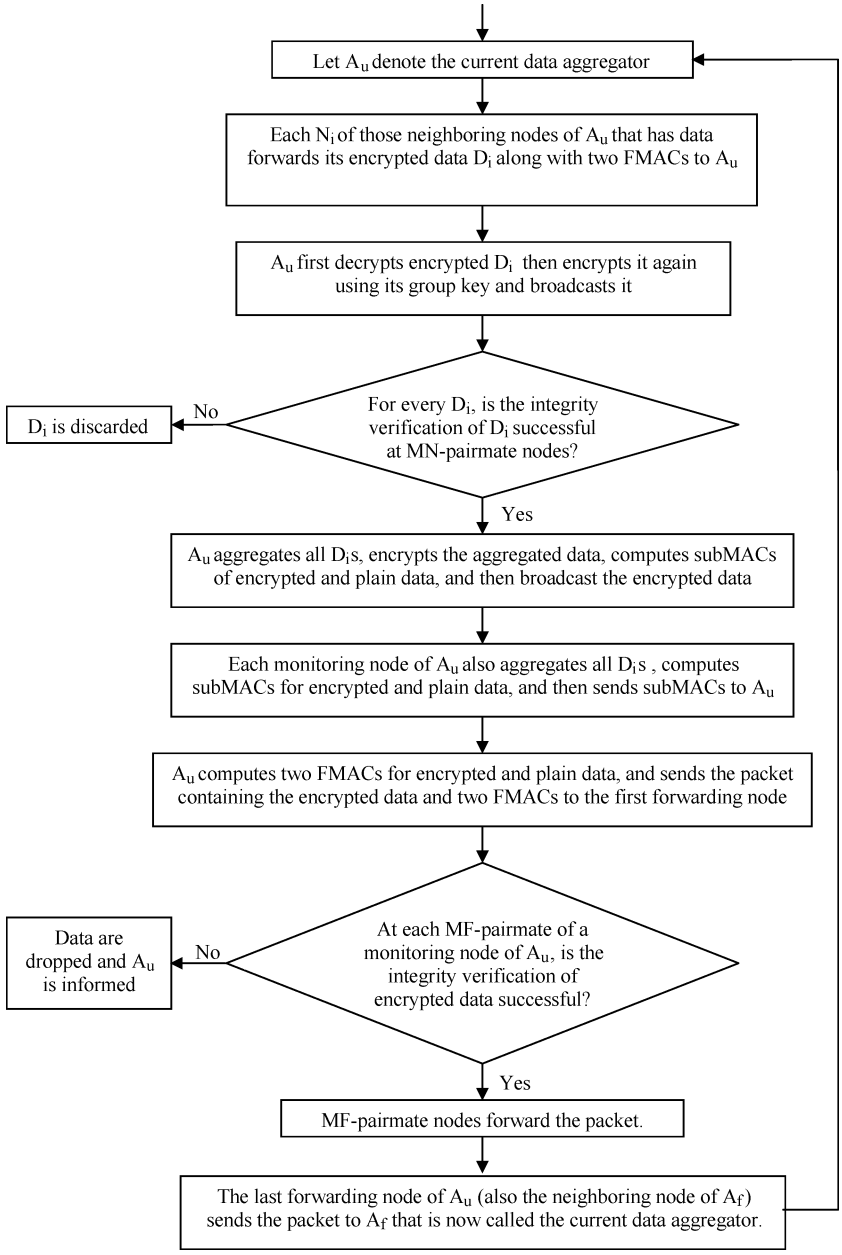

Fig. 7. Flowchart for Algorithm SDFC.

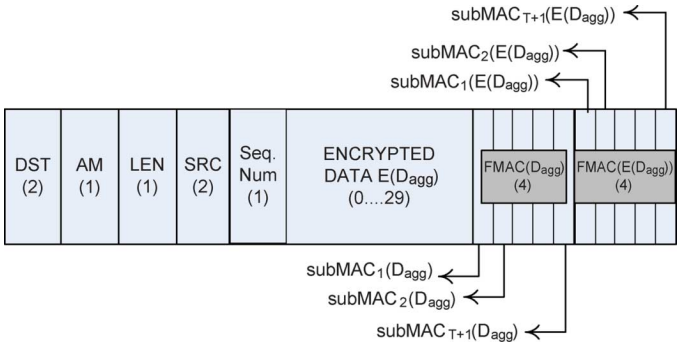

Fig. 8. Packet structure of Algorithm SDFC, where each FMAC is composed of $T+1$ subMACs. The byte size of each field is enclosed in parentheses. The acronyms of the fields are: Dst: destination address; AM: active message type; Len: message length; Src: source address; Seq. Num: packet sequence number.

nodes that serve as forwarding nodes for backward aggregators. In lines 1 to 5 of Algorithm SDFC, the neighboring node N_i sends A_u the data packet $\{E_{K_{bi,u}}(D_i), \mathrm{FMAC}(E_{K_{bi,u}}(D_i)), \mathrm{FMAC}(D_i)\}$ that is originally sent by the backward aggregator A_{bi}. A_u decrypts $E_{K_{bi,u}}(D_i)$ to obtain the plain data D_i. To detect the false data injected by A_{bi} during data aggregation, the data D_i are broadcast, so that those neighboring nodes of A_u that are the MN-pairmates of A_{bi}'s monitoring nodes verify the integrity of the plain data D_i. Recall that A_u and its neighboring nodes share a group key K^u_{group} to provide secure communication between themselves. To protect the confidentiality of D_i, A_u first encrypts D_i using the group key K^u_{group}, and then broadcasts $\{E_{K^u_{\mathrm{group}}}(D_i), \mathrm{FMAC}(D_i)\}$. Those neighboring nodes of A_u that are also the MN-pairmates of A_{bi}'s

Algorithm SDFC

Input: The current data aggregator A_u, the forward data aggregator A_f of A_u, k backward aggregators $\{A_{b1}, \cdots, A_{bk}\}$ of A_u, n neighboring nodes $\{N_1, \cdots, N_n\}$ of A_u for $n \geq T$, T monitoring nodes $\{M_1, \cdots, M_T\}$ of A_u, and z forwarding nodes $\{F_1, \cdots, F_z\}$ of A_u for $z \geq T$.

Output: Any false data, that are injected during data aggregation or forwarding by up to T compromised nodes, are detected and dropped by either A_u's data forwarding nodes or A_f's neighboring nodes. Data confidentiality is provided.

1: **for** (i=1 to k) **do**
2: When the neighboring node N_i of A_u, which also serves as the last forwarding node of the backward aggregator A_{bi}, receives the data $\{E_{K_{bi,u}}(D_i), FMAC(E_{K_{bi,u}}(D_i)), FMAC(D_i)\}$ sent by A_{bi}, the node N_i sends the data to A_u.
3: A_u decrypts $E_{K_{bi,u}}(D_i)$, obtains the plain data D_i, encrypts D_i using the group key K^u_{group}, and finally broadcasts $\{E_{K^u_{group}}(D_i), FMAC(D_i)\}$.
4: All those neighboring nodes of A_u that are the *MN-pairmates* of A_{bi}'s monitoring nodes verify the integrity of D_i. If the verification fails in at least one of these MN-pairmates, A_u is requested to discard D_i, and then A_u and the MN-pairmates of A_{bi} inform A_{bi} about the unsuccessful verification of D_i.
5: **end for**
6: A_u and each monitoring node M_i aggregate all the verified data sent originally by the backward aggregators, for $1 \leq i \leq T$. Let D_{agg} denote the aggregated data.
7: A_u first encrypts D_{agg} using the symmetric key $K_{u,f}$ that it shares with the forward aggregator A_f and then broadcasts the encrypted D_{agg} denoted by $E_{K_{u,f}}(D_{agg})$.
8: Each monitoring node M_i first computes two subMACs: 1) $subMAC(E_{K_{u,f}}(D_{agg}))$ using the key it shares with its *MF-pairmate* forwarding node, and 2) $subMAC(D_{agg})$ using the key it shares with its *MN-pairmate* that is a neighboring node of A_f. Then, each M_i sends its two subMACs to A_u.
9: A_u also computes its own two subMACs, namely $subMAC(E_{K_{u,f}}(D_{agg}))$ and $subMAC(D_{agg})$, using the same key $K_{u,f}$ that it shares with the forward aggregator A_f.
10: A_u forms two full-size MACs, namely $FMAC(D_{agg})$ and $FMAC(E_{K_{u,f}}(D_{agg}))$. $FMAC(D_{agg})$ is formed by concatenating the A_u's $subMAC(D_{agg})$ with the $subMAC(D_{agg})$s of the T monitoring nodes. Similarly, $FMAC(E_{K_{u,f}}(D_{agg}))$ is formed by concatenating the A_u's $subMAC(E_{K_{u,f}}(D_{agg}))$ with the $subMAC(E_{K_{u,f}}(D_{agg}))$s of the T monitoring nodes.
11: A_u first forms a packet containing $E_{K_{u,f}}(D_{agg})$, $FMAC(D_{agg})$ and $FMAC(E_{K_{u,f}}(D_{agg}))$ and, then, sends it to the first forwarding node.
12: **for** (j=1 to z) **do**
13: **if** F_j is not the *MF-pairmate* of a monitoring node of A_u **then**
14: F_j just forwards the incoming packet to the next forwarding node or A_f.
15: **else**
16: F_j verifies the $subMAC(E_{K_{u,f}}(D_{agg}))$ computed by its pairmate of monitoring node.
17: **if** the verification is successful **then**
18: F_j forwards the packet to the next forwarding node if $j < z$. If $j = z$, the forwarding node F_j, which is also a neighboring node of A_f, sends the entire packet to A_f.
19: **else**
20: F_j drops $E_{K_{u,f}}(D_{agg})$ and informs A_u about it.
21: **end if**
22: **end if**
23: **end for**
24: Relabel A_f and A_u as A_u and A_{bz}, respectively, so that the old A_u becomes the z^{th} backward aggregator of the new A_u. Go to Line 1; the z^{th} iteration of the "for" loop (in lines 1 to 8) determines whether D_{agg} contains any false data injected by the old A_u during data aggregation.

Fig. 9. Algorithm SDFC.

monitoring nodes verify the integrity of D_i; if the verification fails in at least one of these MN-pairmates, A_u is requested to discard D_i, and then A_u and the MN-pairmates of A_{bi} inform A_{bi} about the unsuccessful verification of D_i.

A_u and its monitoring nodes aggregate all the verified data to produce the aggregated data D_{agg} (line 6 of Algorithm SDFC). In lines 7 to 10, A_u and its monitoring nodes compute two FMACs of encrypted and plain D_{agg} that are verified by the forwarding nodes of A_u and neighboring nodes of forward data aggregator A_f, respectively. Note that only data aggregators encrypt and decrypt data. For instance, A_u first encrypts D_{agg} using the key $K_{u,f}$ that it shares with its forward data aggregator A_f, and then broadcasts $E_{K_{u,f}}(D_{\text{agg}})$ to its neighbors. Each monitoring node M_i computes one subMAC for the encrypted data $E_{K_{u,f}}(D_{\text{agg}})$ and another subMAC for the plain data D_{agg}. That is, M_i computes one $\text{subMAC}(E_{K_{u,f}}(D_{\text{agg}}))$ using the key that it shares with its MF-pairmate forwarding node, and another $\text{subMAC}(D_{\text{agg}})$ using the key that it shares with its MN-pairmate. A_u also computes its own subMACs of $E_{K_{u,f}}(D_{\text{agg}})$ and D_{agg} using the key $K_{u,f}$ that it shares with A_f. The subMACs of the encrypted data are verified by the MF-pairmates of the monitoring nodes of A_u, whereas the subMACs of the plain data are verified by the MN-pairmates of the monitoring nodes of A_u.

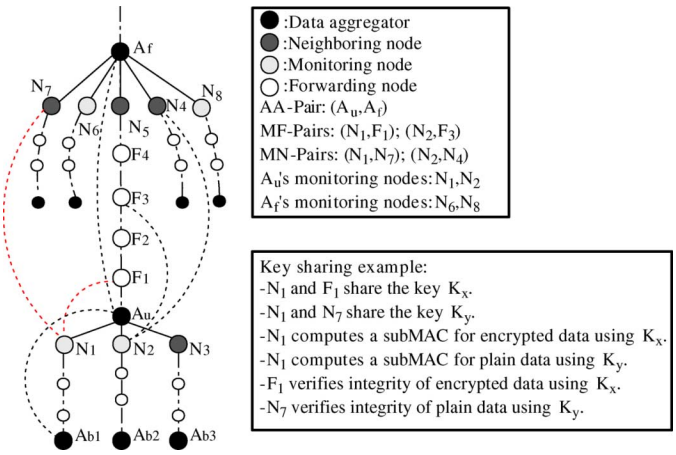

Fig. 10. An example for Algorithm SDFC, for $T = 2$. For the sake of simplicity, some pairs are not illustrated.

After receiving all subMACs of its monitoring nodes, A_u forms two FMACs to transmit along with $E_{K_{u,f}}(D_{\text{agg}})$ (line 10 of SDFC). A_u concatenates the $\text{subMAC}(E_{K_{u,f}}(D_{\text{agg}}))$s of the monitoring nodes with its own $\text{subMAC}(E_{K_{u,f}}(D_{\text{agg}}))$ to form $\text{FMAC}(E_{K_{u,f}}(D_{\text{agg}}))$. Similarly, A_u also concatenates $\text{subMAC}(D_{\text{agg}})$s of the monitoring nodes with its own $\text{subMAC}(D_{\text{agg}})$ to form $\text{FMAC}(D_{\text{agg}})$. A_u finally sends the packet containing $\text{FMAC}(E_{K_{u,f}}(D_{\text{agg}}))$, $\text{FMAC}(D_{\text{agg}})$, and $E_{K_{u,f}}(D_{\text{agg}})$ to the first forwarding node (line 11 of SDFC). During data forwarding (lines 12–23 of SDFC), each of those forwarding nodes that are the MF-pairmates of monitoring nodes verifies the $\text{subMAC}(E_{K_{u,f}}(D_{\text{agg}}))$ computed by its monitoring pairmate. If the verification fails, $E_{K_{u,f}}(D_{\text{agg}})$ is dropped immediately, and A_u is informed about the unsuccessful verification. Otherwise, the forwarding node forwards the data to the next forwarding node. When $E_{K_{u,f}}(D_{\text{agg}})$ arrives at the forward data aggregator A_f, A_f verifies the $\text{subMAC}(E_{K_{u,f}}(D_{\text{agg}}))$ computed by A_u. In line 24, A_f and A_u are relabeled as A_u and A_{bz} respectively, where A_{bz} denotes the zth backward aggregator of the new A_u, for $z \geq 1$. In the zth iteration of the "for" loop in lines 1–5, the new A_u and its those neighboring nodes that are the pairmates of the monitoring nodes of the old A_u determine whether any false data are injected during data aggregation by the old A_u that is the current A_{bz}. To show the basic operations of Algorithm SDFC, we now present an example.

Example 2: Consider the sensor nodes illustrated in Fig. 10 where A_u receives data D_1, D_2, and D_3 from N_1, N_2, and N_3, respectively. Note that D_1, D_2, and D_3 are indeed sent by A_{b1}, A_{b2}, and A_{b3} respectively, and that N_1, N_2, and N_3 are their last forwarding nodes. The neighboring node N_i of A_u sends D_i and its two FMACs to A_u. A_u first decrypts D_i using the symmetric key that it shares with A_{bi}, for $1 \leq i \leq 3$. A_u then encrypts D_1 using the group key K^u_{group} and broadcasts the encrypted D_1 along with the FMAC of plain D_1. If N_1 and N_3 are the MN-pairmates of monitoring nodes of A_{b1}, then the FMAC of plain D_1 that consists of three subMACs need to be verified by A_u, N_1, and N_3. Therefore, N_1 and N_3 decrypt D_1 and verify it using their associated subMACs. Similarly, D_2 and D_3 are also verified by A_u and its neighboring nodes. If any of the MN-pairmates fails to verify any D_i, A_u is requested to discard D_i, and then A_u and the MN-pairmates of A_{bi} inform A_{bi} about the unsuccessful verification of D_i.

Once D_1, D_2 and D_3 are verified, each of A_u and its monitoring nodes N_1 and N_2 aggregates them to obtain the aggregated data D_{agg}. A_u encrypts D_{agg} using the key that it shares with A_f and broadcasts the encrypted D_{agg}. Monitoring node N_1 computes the subMAC for the encrypted D_{agg} using the key that it shares with its MF-pairmate F_1. N_1 also computes the subMAC for the plain D_{agg} using the key that it shares with its MN-pairmate N_7. Similarly, N_2 computes the subMAC for the encrypted D_{agg} using the key that it shares with its MF-pairmate F_3 and computes the subMAC for the plain D_{agg} using the key that it shares with its MN-pairmate N_4. A_u computes two subMACs for the encrypted and plain D_{agg} using the key that it shares with A_f. A_u collects subMACs from N_1 and N_2, forms two FMACs for the encrypted and plain D_{agg}, and finally sends the encrypted D_{agg} along with two FMACs. The FMAC of the encrypted D_{agg} is verified by F_1, F_3, and A_f, whereas the FMAC of plain D_{agg} are verified by N_4, N_7, and A_f.

V. Performance Analysis

In this section, we analyze the performance of DAA. The performance metric used in security analysis is the false data detection ability in the presence of compromised data aggregators and forwarding nodes. The communication and computation overheads are analyzed with respect to additional MAC computations and transmissions required by DAA.

A. Security Analysis of Algorithm SDFC

The security of Algorithm SDFC is analyzed with respect to its false data detection ability. In Algorithm SDFC, compromised nodes can inject false data during data aggregation or data forwarding. When a sensor node is compromised, the intruder is assumed to access all the available security information such as cryptographic keys of the node. We present two lemmas to show that Algorithm SDFC can detect any false data injected by up to T compromised nodes.

The first lemma shows that Algorithm SDFC detects any false data injected by a compromised data aggregator in the process of data aggregation. To be able to distinguish the injected false data from the aggregated data, the monitoring nodes of every data aggregator also perform data aggregation and compute MACs for the aggregated data. The second lemma shows that Algorithm SDFC can also detect any false data injected by forwarding nodes.

Lemma 5.1: Let A_u and A_f denote two consecutive data aggregators, where A_u is the current data aggregator and A_f is the forward data aggregator of A_u. Assume that A_u is compromised and there are additional at most $T-1$ collaborating compromised nodes among the neighboring nodes of A_u and A_f. In Algorithm SDFC, any false data injected by A_u are detected by the A_f's neighboring nodes only.

Proof: See Appendix.

Lemma 5.2: Let A_u and A_f denote two consecutive data aggregators where A_u is the current data aggregator and A_f is the forward data aggregator of A_u. Assume that A_u and A_f are not compromised. Even if all forwarding nodes of A_u are compromised, false data that they inject are detected by A_f.

Proof: See Appendix.

In addition to the attacks by compromised nodes, an intruder can deploy his/her own sensor nodes into the network to inject false data. Such sensor nodes are called *outsider* nodes. Outsider nodes must be prevented from communicating with the network nodes using node authentication techniques. However, even if an outsider node is able to communicate with a network node, the data sent by the outsider node are detected and eliminated by Algorithm SDFC because data aggregators and some forwarding nodes verify the integrity of the forwarded data via subMACs computed by their pairmates. Since the outsider node is not a member of the network, it does not share keys with any forwarding node or data aggregator and cannot provide a valid subMAC for its false data. Therefore, the false data sent by the outsider node are detected by the first forwarding node or data aggregator that performs data verification.

B. Security Analysis of FMAC and subMAC

The security of a MAC scheme can be quantified in terms of the success probability achievable as a function of total number of queries to forge the MAC [21]. The *security* of a z-byte MAC is quantified as $2^{(z\times 8)}$ because an intruder has a 1 in $2^{(z\times 8)}$ chance in blindly forging the MAC. To increase the security of a MAC, its size should be increased. However, increasing the size of the MAC, also increases the communication overhead. To reduce the communication overhead, DAA employs FMACs, each of which is composed of $T+1$ subMACs computed by $T+1$ nodes. Because an FMAC consists of $T+1$ subMACs computed by $T+1$ nodes, we need to answer the question of whether the security of a z-byte FMAC is equivalent to a z-byte MAC computed by a single node.

Let us assume $z = 4$ as the TinySec packet [14] reserves 4 bytes for MAC. Because an FMAC is composed of $T+1$ subMACs, each subMAC has the size of $32/(T+1)$ bits, and the security of a single subMAC is $2^{32/(T+1)}$. Hence, the possibility that the false data are not detected by a subMAC is $1/2^{32/(T+1)}$. In DAA, when an intruder injects false data, the false data need to be verified by T forwarding nodes and the forward data aggregator. Therefore, an intruder can successfully forge a valid FMAC if he/she finds all $T+1$ subMACs with the probability of 1 in $2^{32/(T+1)}$ for each subMAC. Therefore, the probability that the false data are not detected by the FMAC is $(1/2^{32/(T+1)})^{T+1} = 1/2^{32}$. This indicates that a 4-byte FMAC and a 4-byte MAC provide the same security.

The value of T is determined based on security requirements and node density. Because a subMAC cannot be less than 1 bit and an FMAC has $T+1$ subMACs, the maximum value of T is 31 for an FMAC of size 4 bytes. While choosing a large T makes it more difficult for the intruder to launch a false data injection attack, the large T significantly increases the computational overhead as each monitoring node compute MACs of aggregated data.

C. Computational and Communication Cost

In the *traditional data authentication* scheme [14], a source node computes a MAC of its data and sends the data and its MAC to a destination node, which is usually a base station. The destination node checks the integrity of data by verifying the MAC. Hence, false data are detected only by the destination node. Since the previous false data detection techniques [2]–[5] do not address data aggregation and confidentiality, we compare DAA with the traditional data authentication scheme.

1) Computational Cost of Algorithm SDFC: The major computational overhead of DAA occurs due to additional MAC computations in Algorithm SDFC. A data aggregator and its monitoring nodes compute $2\times(T+1)$ subMACs for encrypted and plain data. Because each subMAC of an FMAC is obtained by first computing a MAC and then selecting some bits of it, forming an FMAC requires the computation of $T+1$ MACs. Hence, forming two FMACs requires the computation of $2T+2$ MACs, as opposed to one MAC computation in the traditional data authentication. Moreover, additional $2T+2$ MAC computations are needed to verify all the subMACs of two FMACs by data forwarding nodes, A_f, and the neighboring nodes of A_f. Hence, DAA needs total $4\times(T+1)$ MAC computations. In addition to MAC computations, a data aggregator decrypts and encrypts each data; T monitoring nodes decrypt the broadcasted data and perform data aggregation. The data aggregator also aggregates the data. Therefore, the total computational overhead of DAA is $4\times(T+1)$ MAC computations, $(T+1)$ aggregation processes and $(T+2)$ encryption/decryption processes.

The most expensive operation of MAC computation is the initial key setup phase [22] to produce the cipher context. After the cipher context and the first MAC are produced, the rest of the MAC computations consume much less energy than the first MAC computation. Hence, in DAA, sensor nodes store their cipher context to reduce the MAC computation overhead up to 60% [22]. Given that data transmission consumes much more energy than data computing in wireless sensor networks, DAA compensates the energy increase due to MAC computations by dropping false data as early as possible and reducing the amount of transmitted data using data aggregation, as shown by simulations in Section VI.

2) Communication Cost of Algorithm SDFC: The major communication overhead of DAA occurs as a result of an additional FMAC transmission for false data detection and data transmission during data aggregation in Algorithm SDFC. Because Algorithm SDFC uses two FMACs of 4 bytes each, it has a message overhead of 8 bytes per data packet as opposed to a single MAC of 4 bytes in the traditional data authentication scheme. This implies that the data packet length in DAA can be represented by $(L_{\rm tos}+4)$, where $L_{\rm tos}$ denotes the length (in bytes) of an authenticated and encrypted data packet in TinySec [14]. Let $D_{\rm DAA}$ and $D_{\rm tradAuth}$ denote the amount of data transmitted over a sensor network using DAA and the traditional data authentication scheme, respectively. Next, we show how to compute $D_{\rm DAA}$ and $D_{\rm tradAuth}$.

Let α denote the number of data packets generated by legitimate nodes, and β denote the number of false data packets injected by up to T compromised nodes. Thus, β/α represents the ratio of false data packets to legitimate data packets. Let H_d represent the average number of hops between two consecutive data aggregators, and H represent the average number of hops that a data packet travels in the network. Since DAA detects false data between two data aggregators, false data packets can travel at most H_d hops. In addition, during DAA's data aggregation process, each data is broadcasted by the data aggregator and received by T monitoring nodes, which also send T subMACs to

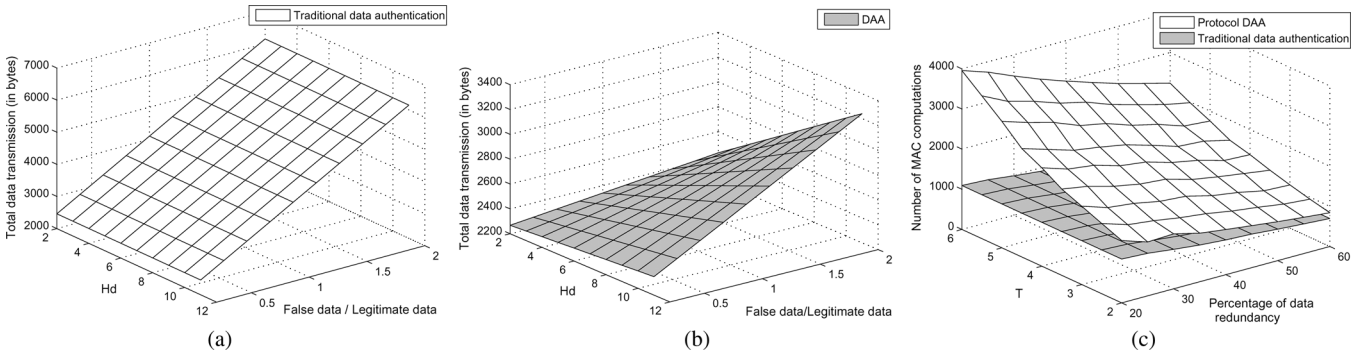

Fig. 11. The total data transmission as a function of average number of hops between data aggregators and the ratio of false data to legitimate data (β/α). Number of compromised nodes is T. If $H_d \leq 12$ and $\beta/\alpha \geq 0.2$, due to elimination of false data, DAA yields less data transmission, hence it saves energy. (a) Network with traditional data authentication scheme (D_{tradAuth}). (b) Network with DAA (D_{DAA}). (c) The overhead of MAC computations versus T for data with variable redundancy.

the data aggregator. However, in the traditional data authentication scheme, false data packets are detected only at base station, and therefore all data packets travel H hops, including false data packets, and there is no data transmission due to data aggregation. Therefore, D_{DAA} and D_{tradAuth} can be expressed as

$$\begin{aligned} D_{\mathrm{DAA}} &= (L_{\mathrm{tos}}+4)\times[(\alpha\times H)+(\beta\times H_d)] \\ &\quad + T\times(L_{\mathrm{tos}}+4)\times(\alpha+\beta) \\ &\quad + T\times\frac{4}{T+1}\times(\alpha+\beta)\ \text{bytes} \\ D_{\mathrm{tradAuth}} &= L_{\mathrm{tos}}\times H\times(\alpha+\beta)\ \text{bytes}. \end{aligned}$$

If D_{DAA} and D_{tradAuth} are normalized by α, then we can write

$$D_{\mathrm{DAA}} = (L_{\mathrm{tos}}+4)\times\left[H+\left(\frac{\beta}{\alpha}\times H_d\right)\right]+\left(1+\frac{\beta}{\alpha}\right) \times\left[T\times(L_{\mathrm{tos}}+4)+\frac{4T}{T+1}\right]\ \text{bytes} \quad (1)$$

$$D_{\mathrm{tradAuth}} = L_{\mathrm{tos}}\times H\times\left(1+\frac{\beta}{\alpha}\right)\ \text{bytes} \quad (2).$$

Although IEEE 802.15.4 standard states that the maximum payload size is 102 bytes, the default size for an encrypted and authenticated TinySec data packet is 41 bytes [14]. After substituting $H = 50$ and $L_{\mathrm{tos}} = 41$ in (1) and (2), the numerical results that are obtained for D_{DAA} and D_{tradAuth} versus H_d and/or β/α are shown in Fig. 11(a) and (b), where D_{tradAuth} increases much faster than D_{DAA}. Fig. 11(a) and (b) also shows that, for $H_d \leq 12$ and $\beta/\alpha \geq 0.2$, the network with DAA transmits less data as compared to the network with traditional data authentication due to elimination of false data during data forwarding. Thus, because DAA detects false data during data forwarding between two consecutive data aggregators and during data aggregation at data aggregators, energy saving of DAA increases as the average number of hops between data aggregators decreases. Also, as β/α increases, the contribution of DAA increases as well. In Fig. 11(a) and (b), the maximum value for β/α is 2, which is assumed to be 9 in [2]. In reality, the amount of injected false data can be much higher than that of legitimate data [2].

VI. Simulation Results

DAA is simulated using QualNet [20] network simulator for an area of 100×100 m^2 and 100 sensor nodes with a transmission range of 15 m. Simulations are performed for both uniform and random distribution of sensor nodes. The base station is located at one corner of the network. Simulations are performed using SNR of 1.5 dB to adapt the high packet loss rate of wireless sensor networks, and the packet retransmission limit is set to 3. Some nodes are designated as data aggregators and distributed into the network area uniformly. Data are assumed to be generated mainly by the nodes located at the edges of the network, although any node is allowed to sense events and generate data. The performance of DAA is compared with the *traditional data authentication* scheme [14], where a source node computes a MAC of its data and sends the data and its MAC to a destination node, which is usually a base station.

A. Computational Overhead

The computational overhead of DAA is evaluated in terms of the number of MAC computations required for false data detection, secure data aggregation, and data confidentiality. Fig. 11(c) compares the number of MAC computations in a network using DAA and traditional data authentication where a source node computes the MAC of the data and its destination node verifies this MAC [14]. The number of MAC computations in the network is shown as a function of security parameter T and percentage of data redundancy. Percentage of data redundancy is defined as being the ratio of redundant data to the total generated data by sensor nodes. The data redundancy is included in the simulations to show the benefit of data aggregation in a network of densely deployed sensor nodes.

Fig. 11(c) illustrates that as T increases, the number of MAC computations in DAA increases as well. Consequently, the network becomes more secure against false data injections because sensor network's ability of detecting and eliminating false data increases. Hence, the value of T trades off between security and computation overhead of the network. Fig. 11(c) also shows that as the percentage of data redundancy increases, the number of MAC computations decreases because data aggregation reduces significantly the amount of data to be transmitted by eliminating redundant data. As seen from Fig. 11(c), DAA has more computational overhead than the traditional data authentication scheme. However, DAA can still result in energy savings because : 1) the data aggregation in DAA significantly reduces the data transmission in the network; and 2) the transmission of a bit can consume as much as energy as the execution of 900 instructions [1].

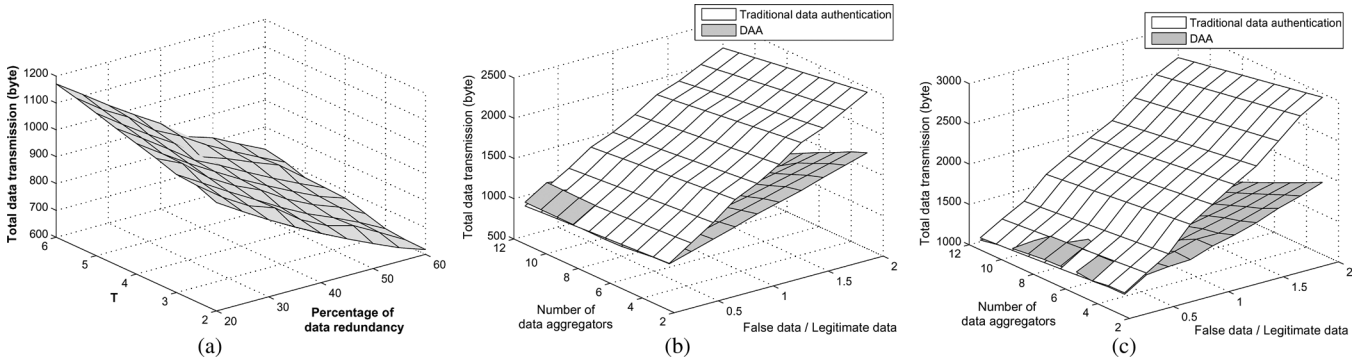

Fig. 12. (a) The impact of T on the communication overhead. (b) Sensor nodes are uniformly distributed. (c) Sensor nodes are randomly distributed. In (b) and (c), the total transmitted data in DAA is compared to that of the traditional data authentication scheme, as the number of data aggregators and the ratio of false data to legitimate data β/α vary. Because DAA reduces the amount of overall data using data aggregation and false data detection, the amount of data transmitted in DAA is up to 60% less than the traditional data authentication scheme.

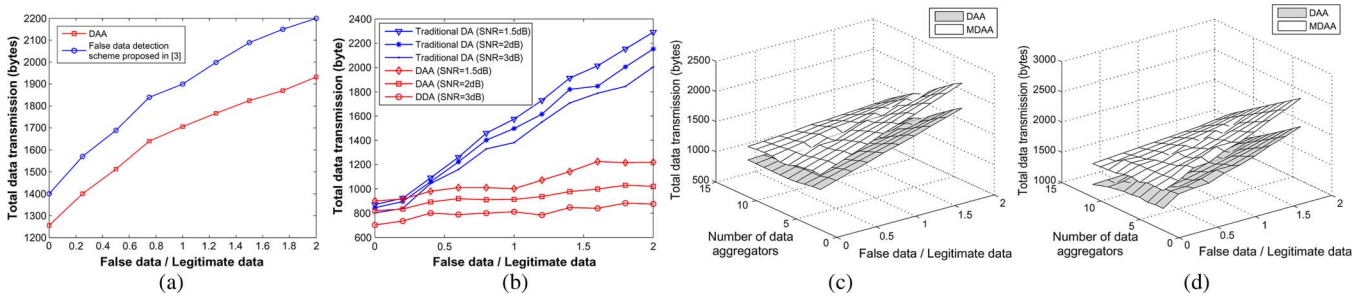

Fig. 13. (a) The total transmitted data in DAA is compared to that of the false data detection scheme proposed in [3]. (b) The total data transmission in the network versus various SNR values. (c) Sensor nodes are uniformly distributed. (d) Sensor nodes are randomly distributed. In (c) and (d), the total transmitted data in DAA is compared to that of the traditional data authentication scheme as the number of data aggregators and the ratio of false data to legitimate data β/α vary. Because DAA reduces the amount of overall data using data aggregation and false data detection, the amount of data transmitted in DAA is up to 60% less than the traditional data authentication scheme.

B. Communication Overhead

The communication overhead of DAA occurs at: 1) the transmission of two FMACs during data forwarding; and 2) those data that are transmitted from data aggregators to their neighboring nodes for aggregation and subMAC computation. Because DAA detects and eliminates false data between two consecutive data aggregators, simulations are performed for various number of data aggregators in the network. The percentage of data redundancy in the network is assumed to be 30% on average. Because most of the energy in sensor networks is consumed due to data transmission, it is critical to mitigate data redundancy and to detect false data as early as possible.

Initially, the impact of T on the communication overhead is evaluated. Although the value of T does not affect the data packet size, it affects the number MAC transmissions between data aggregators and monitoring nodes during data aggregation. Therefore, as it is also shown in the communication analysis, increasing the value of T increases the total data transmission in the network. The simulation is carried out for T values between 2 and 6, and results are presented in Fig. 12(a), which indicates that increasing T from 2 to 6 results in 17% increase in the total data transmission.

The total data transmission of the network with DAA and with traditional data authentication are shown in Fig. 12(b) and (c). When $\beta/\alpha = 2$ and the network has 12 data aggregators, DAA results in 60% less data transmission as compared with the traditional data authentication. This data reduction of up to 60% occurs due to two reasons: 1) the 30% data redundancy is reduced significantly by data aggregation; and 2) those false data that could be twice as much as the legitimate data (i.e., β/α could be equal to 2) are detected and dropped as early as possible. Hence, even though there exists communication overhead, implementing data aggregation and false data detection in DAA still reduces the amount of overall data transmission in the network. As the number of data aggregators increases, the number of hops between data aggregators decreases and DAA detects false data earlier, thereby leading to less data transmission over the network. However, in case of the network with traditional data authentication, the amount of transmitted data is not affected by the number of data aggregators simply because false data are detected only at the base station.

In addition to the traditional data authentication scheme, the communication overhead of DAA is also compared to a well-known false data scheme [3]. The results are presented in Fig. 13(a). In this simulation scenario, 30% data redundancy and six data aggregators are used. As both DAA and [3] require pair forming and key establishment, in the simulation scenario, the data transmission amount includes the overhead due to pair forming and key establishment. Unlike DAA, the false data detection scheme of [3] does not allow data aggregation. Hence, as seen from Fig. 13(a), DAA's communication overhead is less compared to the false data detection scheme proposed in [3]. The communication overhead of DAA is also measured for various SNR values to show the impact of packet loss rate. Fig. 13(b) shows the total data transmission of the network

for different SNR values for both DAA and traditional data authentication. Due to its longer packet length and additional packet transmissions, DAA is affected by packet losses slightly more than traditional data authentication scheme.

C. Impact of Data Aggregation

To show the importance of data aggregation in a densely deployed sensor network, we assume that MDAA is a modified version of DAA such that it is the same as DAA, except that MDAA does not perform any data aggregation at all. That is, DAA mitigates the redundant data at data aggregators by implementing data aggregation, whereas MDAA transmits all of the redundant data to the base station. However, both of them drop false data as soon as it is detected. We compare the performance of DAA with MDAA with respect to the total amount of data transmitted over the network, where β/α ranges from 0.2 to 2 and the number of data aggregators ranges from 2 to 12. The data aggregators are assumed to be distributed uniformly over the network. Simulations also assume that 10 compromised sensor nodes are spread over the network to inject false data and that the percentage of data redundancy is 30% on average. Fig. 13(c) and (d) show that DAA results in up to 25% less data transmission than MDAA. It is worth mentioning that the impact of data aggregation in DAA grows as the percentage of data redundancy increases.

VII. Conclusion

In wireless sensor networks, compromised sensor nodes can distort the integrity of data by injecting false data. Previously known techniques on false data detection do not support data confidentiality and aggregation, even though they are usually essential to wireless sensor networks. However, this paper has presented the novel security protocol DAA to integrate data aggregation, confidentiality, and false data detection. DAA appends two FMACs to each data packet. To reduce the communication overhead of algorithm SDFC, the size of each FMAC is kept fixed. Each FMAC consists of $T+1$ subMACs to safeguard the data against up to T compromised sensor nodes. The performance analysis indicates that the computational and communication overhead of DAA is not substantial, thereby making the implementation of DAA feasible. The simulation results show that the amount of transmitted data is reduced by up to 60%, leading to a significant improvement in bandwidth utilization and energy consumption. As for the future research, we consider of enabling every sensor node to be capable of both aggregating and forwarding data in order to improve network security and efficiency.

Appendix

Proof of Lemma 5.1: We first show how the neighboring nodes of A_f can detect any false data injected by A_u because of the following two reasons. The first reason is that the neighboring nodes of A_u verify all the data broadcast by A_u in line 4 of Algorithm SDFC, so that A_u cannot broadcast its own false data. The second reason is that each of the monitoring nodes also aggregates the entire data by itself (line 9 of Algorithm SDFC) and then computes a subMAC for the aggregated data (line 11 of SDFC). This guarantees that if A_u injects false data, it can be detected by A_f's neighboring nodes that are the MN-pairmates of the monitoring nodes of A_u (lines 1 to 8 of Algorithm SDFC). Since the subMACs of the plain aggregated data are verified by T neighboring nodes of A_f, A_u needs at least T compromised monitoring nodes to inject false data.

Now, we show that A_f and the forwarding nodes of A_u cannot detect the false data injected by A_u. A_f cannot detect the false data because it verifies the subMAC computed by A_u. If A_u is compromised, A_u can construct the subMAC after the false data are injected and, therefore, A_f cannot detect the false data by verifying the same subMAC. Similarly, after injecting false data and encrypting it, A_u can broadcast it and request the monitoring nodes to send subMACs for the encrypted data. Because those forwarding nodes of MF-pairmate verify the encrypted data, their verification becomes successful and, therefore, the false data are not detected by the forwarding nodes.■

Proof of Lemma 5.2: In Algorithm SDFC, those forwarding nodes that are the MF-pairmates of A_u's monitoring nodes verify the encrypted data (lines 15 to 26 of Algorithm SDFC), but they do not compute new subMACs for the verified data. Therefore, a compromised forwarding node that injects false data cannot add a new subMAC for its false data. Because all the forwarding nodes of A_u are assumed to be compromised in the lemma, the false data injected by these compromised nodes are not detected during data forwarding. This implies that A_f receives the false data. In lines 1 to 8 of Algorithm SDFC, data aggregators verify the received data using the subMACs computed by the backward aggregators. Therefore, when A_f receives the false data from compromised forwarding nodes, A_f fails to verify the subMAC computed by A_u, which is assumed to be not compromised. It follows that the lemma holds.■

Acknowledgment

The authors are grateful to Mr. D. H. An for having discussions in developing Algorithm MNS and the security analysis of FMAC. The authors also sincerely thank anonymous reviewers for their comments that helped improve the quality of the paper.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Commun. Mag.*, vol. 40, no. 8, pp. 102–114, Aug. 2002.

[2] F. Ye, H. Luo, S. Lu, and L. Zhang, "Statistical en-route detection and filtering of injected false data in sensor networks," in *Proc. IEEE INFOCOM*, 2004, vol. 4, pp. 2446–2457.

[3] S. Zhu, S. Setia, S. Jajodia, and P. Ning, "Interleaved hop-by-hop authentication against false data injection attacks in sensor networks," *ACM Trans. Sensor Netw.*, vol. 3, no. 3, Aug. 2007.

[4] H. Yang and S. Lu, "Commutative cipher based en-route filtering in wireless sensor networks," in *Proc. IEEE VTC*, 2004, vol. 2, pp. 1223–1227.

[5] Z. Yu and Y. Guan, "A dynamic en-route scheme for filtering false data in wireless sensor networks," in *Proc. IEEE INFOCOM*, Barcelona, Spain, Apr. 23–27, 2006, pp. 1–12.

[6] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, "Impact of network density on data aggregation in wireless sensor networks," in *Proc. 22nd Int. Conf. Distrib. Comput. Syst.*, Jul. 2002, pp. 575–578.

[7] A. Perrig, R. Szewczyk, D. Tygar, V. Wen, and D. Culler, "SPINS: Security protocols for sensor networks," *Wireless Netw. J.*, vol. 8, pp. 521–534, Sep. 2002.

[8] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. IT-22, no. 6, pp. 644–654, Nov. 1976.

[9] L. Hu and D. Evans, "Secure aggregation for wireless networks," in *Proc. Workshop Security Assurance Ad hoc Netw.*, Orlando, FL, Jan. 28, 2003, pp. 384–394.

[10] B. Przydatek, D. Song, and A. Perrig, "SIA: Secure information aggregation in sensor networks," in *Proc. SenSys*, 2003, pp. 255–265.

[11] H. Çam, S. Ozdemir, P. Nair, D. Muthuavinashiappan, and H. O. Sanli, "Energy-efficient and secure pattern based data aggregation for wireless sensor networks," *Comput. Commun.*, vol. 29, no. 4, pp. 446–455, Feb. 2006.
[12] 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 GLOBECOM*, 2003, pp. 1435–1439.
[13] "Crossbow Technologies Inc.," [Online]. Available: http://www.xbow.com
[14] C. Karlof, N. Sastry, and D. Wagner, "TinySec: A link layer security architecture for wireless sensor networks," in *Proc. 2nd ACM Conf. Embedded Netw. Sensor Syst.*, 2004, pp. 162–175.
[15] D. Seetharam and S. Rhee, "An efficient pseudo random number generator for low-power sensor networks," in *Proc. 29th Annu. IEEE Int. Conf. Local Comput. Netw.*, 2004, pp. 560–562.
[16] W. Du, J. Deng, Y. S. Han, and P. K. Varshney, "A pairwise key predistribution scheme for wireless sensor networks," in *Proc. 10th ACM CCS*, 2003, pp. 42–51.
[17] D. Liu, P. Ning, and R. Li, "Establishing pairwise keys in distributed sensor networks," *ACM Trans. Inf. Syst. Security*, vol. 8, no. 1, pp. 41–77, Feb. 2005.
[18] C. Blundo, A. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly-secure key distribution for dynamic conferences," in *Proc. Crypto*, 1992, pp. 471–486.
[19] S. Xu, "On the security of group communication schemes based on symmetric key cryptosystems," in *Proc. ACM Workshop Security Ad hoc Sensor Netw.*, 2005, pp. 22–31.
[20] "QualNet network simulator," Scalable Network Technologies [Online]. Available: www.scalable-networks.com/
[21] P. Gauravaram, W. Millan, J. G. Nieto, and E. Dawson, "3C—A provably secure pseudorandom function and message authentication code: A new mode of operation for cryptographic hash function," Cryptology ePrint archive, Rep., 2005.
[22] R. L. Rivest, "The RC5 Encryption Algorithm," in *Proc. 2nd Int. Workshop FSE*, 1994, pp. 86–96.
[23] M. Sivrianosh, D. Westhoff, F. Armknecht, and J. Girao, "Non-manipulable aggregator node election protocols for wireless sensor networks," in *Proc. IEEE WiOpt*, Cyprus, Apr. 2007, pp. 1–10.
[24] J. Newsome, E. Shi, D. Song, and A. Perrig, "The Sybil attack in sensor networks: Analysis and defenses," in *Proc. 3rd IEEE/ACM IPSN*, 2004, pp. 259–268.
[25] R. Rajagopalan and P. K. Varshney, "Data aggregation techniques in sensor networks: A survey," *IEEE Commun. Surveys Tutorials*, vol. 8, no. 4, 4th Quarter 2006.
[26] J. Deng and Y. S. Han, "Using MDS codes for the key establishment of wireless sensor networks," in *LNCS 3794*. Berlin, Germany: Springer-Verlag, 2005, pp. 732–744.
[27] Q. Dong and D. Liu, "Using auxiliary sensors for pairwise key establishment in WSN," in *Proc. IFIP Int. Conf. Netw.*, 2007, pp. 251–262.
[28] K. Wu, D. Dreef, B. Sun, and Y. Xiao, "Secure data aggregation without persistent cryptographic operations in wireless sensor networks," *Ad Hoc Netw.*, vol. 5, no. 1, pp. 100–111, 2007.

Suat Ozdemir (M'03) received the M.Sc. degree in computer science from Syracuse University, Syracuse, NY, in 2001, and the Ph.D. degree in computer science from Arizona State University, Tempe, respectively.

He has been with the Computer Engineering Department, Gazi University, Ankara, Turkey, since March 2007. His main research interests include broad areas of wireless networks and network security.

Dr. Ozdemir serves on the Technical Program Committee for several conferences such as ICC, IEEE GLOBECOM, etc. He also serves as a reviewer for several journals, e.g., the IEEE TRANSACTIONS ON MOBILE COMPUTING, the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE/ACM TRANSACTIONS ON NETWORKING, and the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY.

Hasan Çam (SM'01) received the B.S. and M.S. degrees in electrical engineering from Istanbul Technical University, Istanbul, Turkey, in 1982 and 1984, respectively; the M.S. degree in computer science from Polytechnic University, New York, NY, in 1986; and the Ph.D. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, in 1992.

He is a Senior Research Scientist with Altusys Corporation, Trenton, NJ. He has previously served as a faculty member at Arizona State University, Tempe; University of Arkansas, Fayetteville; and King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia. He has one U.S. patent pending. His current research interests include wireless sensor and cellular networks, body area networks, network security, low-power interconnection, and processor architectures.

Dr. Çam was a member of the Editorial Boards of *Computer Communications* (from 2001 to 2009) and the *International Journal of Communication Systems* (from 2001 to 2007). He served as a Guest Editor on *Computer Communications*. He also organized symposiums and served as a Technical Program Committee member in conferences.