SEDAN: Secure and Efficient Data Aggregation protocol for wireless sensor Networks

Miloud Bagaa, Noureddine Laslaa, Abdelraouf Ouadjaout, Yacine Challal

SEDAN: Secure and Efficient protocol for Data Aggregation in wireless sensor Networks

Miloud Bagaa*, Noureddine Lasla†, Abdelraouf Ouadjaout*, Yacine Challal‡
*University of Science and Technology Houari Boumediene, LSI lab
Algiers, Algeria
Email: {bagmoul,ouadjaout}@gmail.com
†Institut National de formation en Informatique
Algiers, Algeria
Email: lnoureddine4@gmail.com
‡Compiegne University of Technology, Heudiasyc lab
Compiegne, France
Email: ychallal@hds.utc.fr

***Abstract*—Energy is a scarce resource in Wireless Sensor Networks. Some studies show that more than 70% of energy is consumed in data transmission. Since most of the time, the sensed information is redundant due to geographically collocated sensors, most of this energy can be saved through data aggregation. Furthermore, data aggregation improves bandwidth usage. Unfortunately, while aggregation eliminates redundancy, it makes data integrity verification more complicated since the received data is unique.**

In this paper, we present a new protocol that provides secure aggregation for wireless sensor networks. Our protocol is based on a two hops verification mechanism of data integrity. Our solution is essentially different from existing solutions in that it does not require referring to the base station for verifying and detecting faulty aggregated readings, thus providing a totally distributed scheme to guarantee data integrity.

We carried out simulations using TinyOS environment. Simulation results show that the proposed protocol yields significant savings in energy consumption while preserving data integrity.

I. Introduction

In recent years, advances in wireless communication and embedded systems allowed the development of low-cost and low-power wireless sensor nodes. *Wireless Sensor Networks* (WSN) represent an emerging research area, providing useful applications in various fields such as habitat monitoring, battlefield surveillance and forest fire monitoring.

Node's primary function is to gather measurement data from the environment, and collaborate with other sensors to route them to a processing centre, which is also called the base station or sink node. Such sensors are generally equipped with low energy supply and small storage capability. Therefore, any protocol design must consider these constraints by providing resource conserving solutions [1].

To improve fault tolerance and sensing quality, one solution is to increase data redundancy by deploying sensors with high density. Nevertheless, this redundancy will cause significant power overhead and increasing collisions. To overcome these problems, many researches have been focused on *aggregating sensed data*, using different mathematical functions: such as MAX, MIN, SUM... *etc*.

The data aggregation paradigm is an essential key for the lifetime of the network due to the reduced number of broadcasts and collisions. However, data aggregation is potentially vulnerable to attackers who may inject bogus information or forge aggregated values without being detected.

At the aggregation layer, many security services can be provided. In this paper, we focus on the integrity of the aggregated data, due to its importance in sensing applications.

Many solutions, including several mechanisms to secure data aggregation, have been proposed [2], [3], [4], [5]. These solutions fall into two main categories: *hop by hop* and *end to end* protocols. In hop by hop approach, integrity verification is carried out by sensor nodes with the help of the sink. In end to end protocols, the base station is the only responsible of the overall verification mechanism [6].

The main drawbacks of existing solutions are the *centralization* and the *blind rejection*.

The centralization problem is due to the total or partial rely on the sink for the verification process. Therefore, the application is not totally distributed, and no verification can be done before the arrival of all aggregated data to the sink.

The second important problem is the blind rejection. When a malicious node is detected, the sink must reject the received aggregation value. Thus, an important amount of correct data is lost.

In this paper, we present a new secure aggregation protocol solving the above problems. Our work is based on two hops verification mechanism that is inspired from SAWN (Secure Aggregation for Wireless Network) [2]. However, SAWN is not immune to the weaknesses described above. In our solution, called SEDAN (for *Secure and Efficient Data Aggregation protocol for wireless sensor Networks*), each node can verify *immediately* the integrity of its two hops neighbors' data, and the aggregation of the immediate neighbors. This improvement allows avoiding useless transmission of bogus data, and hence saving sensors' energy resources.

The cornerstone of our solution is the management of a new type of key called *two hops pair-wise key*. This new key allows

sharing a secret between any two hops neighbors, unknown by the intermediate node. This way, any sensor can verify the data integrity of its two hops neighbors, and insure that it was not modified by the intermediate node.

The rest of this paper is organized as follows. Related works are presented in section II.The design goals of SEDAN and the protocol details are presented in section III. In section IV, we describe the Extended Pair-wise Key Establishment protocol, for establishing all needed key materials. Section V provides a security analysis of our protocol SEDAN. The overall performance evaluation and simulation results are detailed in section VI. We draw conclusions in section VII.

II. Related Works

The aggregation paradigm aims to replace individual readings of single nodes by a collaborative view of specific areas. This collaboration induces that many nodes should participate in the calculation of the final result, and enables aggregator nodes to manipulate any received data of their region.

However, in real deployment, a network can contain intruders or compromised nodes. In such situations, the basic aggregation mechanism is very vulnerable to different threats including modification, injection ... *etc*.

Hence, it is very important to verify the correct behavior of the aggregator nodes, and prevent them to falsify the real collaborative view.

A. Existing solutions

Kui et al [5] proposed a protocol based on a clique infrastructure. Nodes in a same clique can listen to each others, forming a complete graph. The protocol uses the watch dog method to verify the calculated aggregation value of the parent.

In [3], Chan et al proposed a protocol based on commitment tree. The verification of aggregation value occurs at leaf nodes level. The base station sends a proof of the final aggregation to leaf nodes. Each leaf node reconstitutes the intermediate aggregation values of each hop toward the sink, using its *off-path* vertices. The final result is compared to the sink's proof.

Yang et al [7] proposed SDAP. The protocol uses a *divide-and-conquer* approach. It is based on a probabilistic technique to partition the nodes into multiple sub trees. Data is aggregated within each sub tree using a hop by hop verification. Root nodes send their aggregation value to the base station using shared secret keys.

Mahimkar and Rappaport in [8] present a protocol based on a cluster structure. Each cluster-head sends to the sink the average of its members' data signed by all of them using elliptique curve cryptography.

SAWN [2] is a hop by hop secure aggregation protocol, based on a *two hops verification mechanism*. It proposes a novel solution for a node to prevent data modification by the next hop during the aggregation. Each node generates a special proof verifiable by the two hops parent. This proof will allow the comparison between the calculated aggregation and the original data. In the next section, we give a brief description of SAWN and analyze the main drawbacks of its solution.

Our solution is an improvement over SAWN that resolves the centralization and the blind rejection problems.

B. Terminology and Notation

In table I we introduce the different notations and terminology used throughout this paper to describe the different solutions and our own protocols.

Notation	Description
ID_A	Identifier of node A
BS	The base station
d_A	Data of node A. It can represent the reading of A or its aggregated data
N_A	Nonce generated by A
f	An aggregation function
K_A	Current round key of node A in SAWN protocol
$MAC(K, m)$	Authetication message code of m using key K
$E(K, m)$	Encryption of m using key K
$A \rightarrow B : m$	A sends to B the message m
$K_{A,B}$	Secret pair wise key between A and B. A can be a one hop or a two hops neighhbor of B
$\|\|$	Concatenation
MK_A	Master key of node A
G	A one way function
$P(A)$	Parent of A
$GP(A)$	Grandparent of A

TABLE I
Notations

C. SAWN

SAWN was the first protocol to introduce the two hops verification mechanism. The main assumption of the protocol is that two consecutive nodes can not be compromised.

In addition to the two hops verification mechanism, SAWN is based on the concept of *delayed verification*, which leads to some important drawbacks.

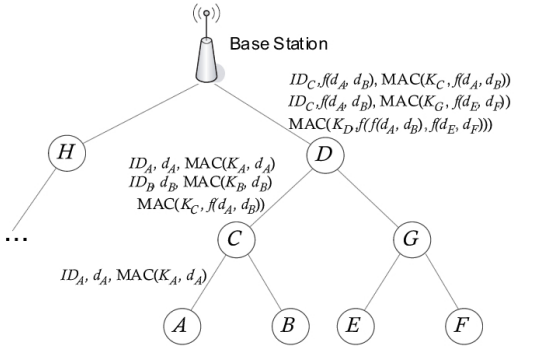


Fig. 1. Aggregation process in SAWN

Figure 1 explains the aggregation process in SAWN. At each round r, each leaf node of the tree sends its measurement reading and a Message Authentication Code (MAC) using its secret key of the current round:

$$A \rightarrow C : ID_A, d_A, MAC(K_A, d_A)$$

When the parent receives its child's message, it saves it. Upon receiving all children's data, the parent calculates the

aggregation of the received data, generates the MAC and sends it with all saved messages to the next parent:

$$C \rightarrow D: \quad \begin{array}{l} ID_A, d_A, MAC(K_A, d_A) \\ ID_B, d_B, MAC(K_B, d_B) \\ MAC(KC, f(d_A, d_B)) \end{array}$$

The grandparent stores all received messages, and calculates the aggregation value of each child. For example, node D having all grandchildren's data can calculate the aggregation of C and G. In addition, it calculates the MAC of the overall aggregation.

$$D \rightarrow BS: \quad \begin{array}{l} ID_C, f(d_A, d_B), MAC(K_C, f(d_A, d_B)) \\ ID_G, f(d_E, d_F), MAC(K_G, f(d_E, d_F)) \\ MAC(K_D, f(f(d_A, d_B), f(d_E, d_F)) \end{array}$$

After receiving all final aggregation results, the base station starts the verification process. It reveals nodes' keys to the entire network. To achieve this goal, SAWN supposed a powerful base station capable to reach the entire network by one global broadcast. This revelation of keys enables each node to verify grandchildren's data integrity and the aggregation of each child. Note that the keys are authenticated using μTESLA [9] protocol.

However, this solution is not scalable to large networks that may contain thousands of nodes. Indeed, this referring to the base station during the verification process will cause a significant delay. Furthermore, sending authentication keys of the entire network requires the use of multiple packets. To analyze the overhead of this solution, we consider the following scenario. The revelation of a node's key requires 6 bytes : 2 bytes for the identifier and 4 bytes for the key. Each packet must contain also a MAC (4 bytes) using the μTESLA key. If we consider using TinyOS [10], the default packet size is 29 bytes. So, each packet can transmit only the revelation of at most 4 nodes.

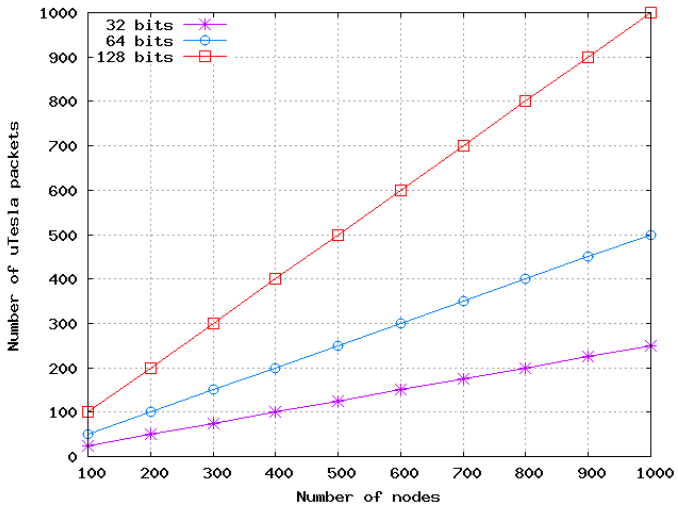


Fig. 2. Number of required μTesla packets vs. number of nodes

Figure 2 illustrates the variation of the number of messages for different packet sizes, by varying the size of the transmitted keys. We remark that the solution of SAWN consumes a considerable number of packets, which grows linearly relatively to the network size ($O(n)$).

Another important weakness of SAWN is the amount of rejected data. In SAWN, and many other protocols [3], [11], [4], the violation of data integrity at any place in the network obligates the sink to reject the received aggregation data. Since this latter represents the view of the whole *infected branch*[1], this rejection yields to a significant data loss.

Moreover, since nodes are responsible of data verification and this phase can not be started before the key revelation (which leads to considerable delays in large networks), the base station should wait for a certain period of time before committing any received data.

III. Our Solution: SEDAN

A. Design Goals

To overcome the above constraints, our solution bypasses the need of the general broadcast by introducing a new type of key that is shared between two hops neighbors. This key must be kept secret from neighboring nodes. When a node transmits its data or its aggregation value, it calculates a MAC using its two hops pair-wise key shared with the grandparent. Since this key is unknown by the next hop, the integrity of the data is preserved and any modification in the aggregation value can be detected by the grandparent.

Therefore, SEDAN enables a *distributed* and *in-network verification scheme* scalable to large networks.

Besides, SEDAN blocks (without any delay) the tempered data at the compromised node level, avoiding transmitting it to the sink and infecting other correct aggregation values. Hence, all aggregation results arriving to the base station are correct and can be committed *immediately*.

B. Assumptions

- The aggregation function should verify the following relation [11]:

  $$f(d_1, d_2, d_3, ..., d_n) = f(d_1, f(d_2, d_3, d_4), d_5, ..., d_n)$$

  This means that the final result could be computed with any combination of sub results using the same function. Many important aggregation functions verify this property like MIN, MAX, AVERAGE ... *etc*.
- In the description of SEDAN, we assume a tree communication topology, but the proposed protocol can be used with any other hierarchical structure.
- We maintain the SAWN assumption: the tree should not contain two consecutive compromised nodes.

C. Description

SEDAN is a lightweight and secure protocol consisting of the following steps :

[1]The infected branch is the tree containing the malicious node and having a sink's neighbor as root.

1) Key establishment: The initial step consists of establishing all needed pair-wise keys. Note that this execution is required only once during the network lifetime, in contrast to the aggregation protocol. However, to guarantee strong security, keys should be refreshed periodically.

Each node needs to establish four types of pair-wise keys:

- A one hop pair-wise key with its parent.
- A one hop pair-wise key with each child.
- A two-hop pair-wise key with its grandparent.
- A two hop pair-wise key with each grandchild.

The details of establishment of these keys are given in the next section.

2) Data authentication: When a node i wants to send its data d_i, it sends to its parent the following packet:

$$\begin{aligned} &ID_i, N_i, d_i, \\ &MAC(K_{i,P(i)}, N_i \mid\mid d_i), \\ &MAC(K_{i,GP(i)}, d_i \mid\mid N_i) \end{aligned}$$

The nonce N_i allows preventing from replay attacks that reinject the same data. The first MAC is called a *One Hop MAC* (OHM). It is computed using the pair-wise key shared with the parent, and enables this latter to check the packet's origin. This verification defends against impersonation attacks.

The last MAC is called a *Two Hops MAC* (THM), calculated using the pair-wise key shared with the grandparent. The THM allows performing the two hops verification mechanism, as described in step 5.

3) One-hop data integrity verification: When a node j receives a data packet from a child, it verifies the OHM to validate the origin of the packet, and stores the rest of information: ID_{child}, N_{child}, d_{child} and THM_{child}.

4) Authentication of aggregated data: Upon receiving the data of all its children, the node j computes the aggregation value over the children's data.

Since $f(d_{child_1}, \ldots, d_{child_n})$ will represent the data of j, the node must calculate its OHM :

$$MAC(K_{j,P(j)}, N_j \mid\mid f(d_{child_1}, \ldots, d_{child_n}))$$

and its THM :

$$MAC(K_{j,GP(j)}, f(d_{child_1}, \ldots, d_{child_n}) \mid\mid N_j)$$

Then, node j transmits to its parent :

$$\begin{aligned} &ID_{child_1}, N_{child_1}, d_{child_1}, THM_{child_1} \\ &\vdots \\ &ID_{child_n}, N_{child_n}, d_{child_n}, THM_{child_n} \\ &ID_j, Nonce_j, \\ &MAC(K_{j,P(j)}, N_j \mid\mid f(d_{child_1}, \ldots, d_{child_n})) \\ &MAC(K_{j,GP(j)}, f(d_{child_1}, \ldots, d_{child_n}) \mid\mid N_j) \end{aligned}$$

5) Two-hops data integrity verification: At the reception of an aggregation packet from j, a node k must verify the aggregation behavior of j:

- To guarantee correct relaying of grandchildren's data, node k verifies the THM of each one of them. Furthermore, node k reconstitutes the correct aggregation value of j using the grandchildren's data.
- Having the real aggregation, node k can compare it to the aggregation performed by j. Thus, SEDAN can detect faulty aggregation immediately without any delay, and all bogus data are stopped to preserve the correctness of branch's aggregation value.

Node k considers the reconstituted aggregation value of each child as the child's data, and reiterate the step 4 by sending:

$$\begin{aligned} &ID_{child_1}, N_{child_1}, d_{child_1}, THM_{child_1} \\ &\vdots \\ &ID_{child_n}, N_{child_n}, d_{child_n}, THM_{child_n} \\ &ID_k, N_k, \\ &MAC(K_{k,P(k)}, N_k \mid\mid f(d_{child_1}, \ldots, d_{child_n})) \\ &MAC(K_{k,GP(k)}, f(d_{child_1}, \ldots, d_{child_n}) \mid\mid N_k) \end{aligned}$$

Note that if an aggregator node j wants to send its own measurement data within an aggregation packet, it must transmit it with an extra $OHM = MAC(K_{j,P(j)}, N_j \mid\mid d_j)$. Node j sends also its THM on the aggregation value over its reading and children's data.

$$\begin{aligned} &ID_{child_1}, N_{child_1}, d_{child_1}, THM_{child_1} \\ &\vdots \\ &ID_{child_n}, N_{child_n}, d_{child_n}, THM_{child_n} \\ &ID_j, N_j, d_j, MAC(K_{j,parent}, N_j \mid\mid d_j) \\ &MAC(K_{j,P(j)}, N_j \mid\mid f(d_{child_1}, \ldots, d_{child_n}, d_j)) \\ &MAC(K_{j,GP(j)}, f(d_{child_1}, \ldots, d_{child_n}, d_j) \mid\mid N_j) \end{aligned}$$

The next hop k verifies all the grandchildren's THM and the OHM of j. Hence, node k can recalculate the aggregation of node j and forwards the required packets as in step 6.

D. Example

Figure 3 illustrates the messages exchanged during an aggregation using our protocol SEDAN.

- Node *A* sends the measurement data and its OHM and THM :

$$\begin{aligned} A \rightarrow C: \quad & ID_A, N_A, d_A, \\ & MAC(K_{A,C}, N_A \mid\mid d_A) \\ & MAC(K_{A,D}, d_A \mid\mid N_A) \end{aligned}$$

- Node B doing the same, C receives two data packets. C aggregates all readings and sends the following packet:

$$\begin{aligned} C \rightarrow D: \quad & ID_A, N_A, d_A, MAC(K_{A,D}, d_A \mid\mid N_A) \\ & ID_B, N_B, d_B, MAC(K_{B,D}, d_B \mid\mid N_B) \\ & ID_C, N_C, MAC(K_{C,D}, N_C \mid\mid f(d_A, d_B)) \\ & MAC(K_{C,BS}, f(d_A, d_B) \mid\mid N_C) \end{aligned}$$

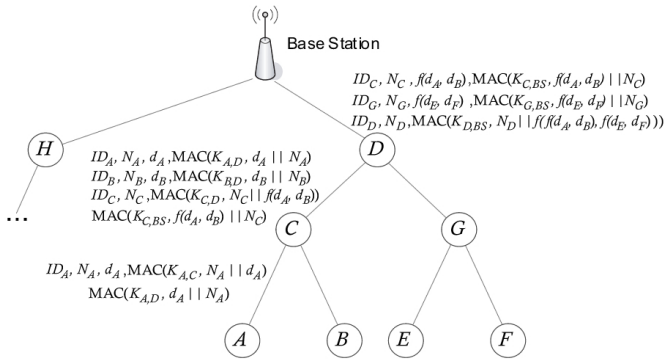


Fig. 3. Aggregation process in SEDAN

- Node D can verify the original data of A and B, and the computed aggregation of C. The same mechanism is applied to message sent by G.

$$\begin{aligned} D \rightarrow BS: \quad & ID_C, N_C, f(d_A, d_B), \\ & MAC(K_{C,BS}, f(d_A, d_B) \mid\mid N_C) \\ & ID_G, N_G, f(d_E, d_F) \\ & MAC(K_{G,BS}, f(d_E, d_F) \mid\mid N_G) \\ & ID_D, N_D, \\ & MAC(K_{D,BS}, N_D \mid\mid \\ & \quad f(f(d_A, d_B), f(d_E, d_F))) \end{aligned}$$

IV. EPKE : Extended Pair-wise Key Establishment protocol

There exists considerable work [12], [13], [14], [15] dealing with the problem of establishing a secure communication infrastructure within a network of sensor nodes. The appropriate scheme to WSN should use symmetric-key and organize itself without prior knowledge of deployment.

Due to the absence of a fixed infrastructure, the limited computation abilities and resources of sensor nodes, the pre-distribution keying protocols are more suitable to such environments. In these protocols, all nodes are preloaded with secret information that will be used to establish pair-wise keys between neighboring nodes.

The pre-distribution keying protocol in [14] is based on a probabilistic approach. Each node carries *k* distinct keys that are randomly chosen from a large key pool. A pair of nodes can establish a pair-wise key, if they can find one common key within their subsets of keys. The drawback of this approach is that it requires more memory for storing keys in large scale networks.

Authors in [12], [13] proposed a family of protocols for establishing a pair-wise key based on the *transitory initial keys* (TIK) concept. In these protocols, the same transitory initial key is pre-configured into each sensor node. A node uses this key to generate a pair-wise key to share with each of its neighbors. After the key setup phase, each node *erases* the initial key from its EEPROM memory. The main supposition behind the TIK concept is that an intruder can not obtain the initial key by compromising a legitimate node during the key setup phase. This period of time is supposed short and represents the minimum time ($T_{\min}$) needed by a node to establish keys with its neighbors.

In this section, we propose a solution to establish the two hops and one hop pair-wise keys, based on the transitory initial key (TIK) setup scheme of LEAP [12] and OTMK [13].

A. Initialization

Before the deployment, sensors are preloaded with a transitory initial key K_{IN}. Each node u drives its master key MK_u,

$$MK_u = G(K_{IN}, ID_u)$$

As the node is deployed in the network, the initial key is used to establish a pair-wise key with each neighbor. Every initial key is only valid for a time $T_{\min}$. After this time, every node will erase the initial key K_{IN}.

B. One hop pair-wise key establishment

This kind of keys is used in SEDAN to compute OHM. This allows building a secure link between neighboring nodes and defending against impersonation attacks.

After nodes are deployed, each one discovers its neighbors and tries to establish a one hop pair-wise key with them. Depending on whether the node's neighbor erased its initial key or not, we distinguish two cases:

1) Case 1: When two neighbors still have K_{IN}, they use it to compute a shared key. To obtain a symmetric key between two nodes u and v, we use the following formula:

$$K_{u,v} = G(MK_{min(u,v)}, max(u,v) \mid\mid N_{max(u,v)}) \quad (1)$$

Note that u and v can compute $MK_{min(u,v)}$ because each one knows K_{IN} and can generate the master key of any other node.

Figure 4a describes the different steps to establish a one hop pair-wise key $K_{u,v}$ between u and a neighbor v within $T_{\min}$.

Note that node v will send also a $Join1$ message to u. After the above steps, node u will have established a one hop pair-wise key with each of its immediate neighbors in initialization phase.

2) Case 2: After $T_{\min}$, node erases the transitory initial key K_{IN}, and will not be able to generate other master keys. So, when a new node u is initialized, each neighbor v, having erased K_{IN}, should use its own master key to generate a pair-wise key with u (since u can generate any master key) :

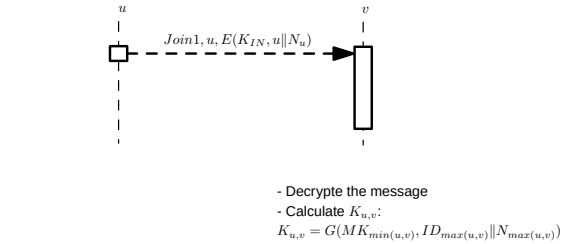
$$K_{u,v} = G(MK_v, ID_u \mid\mid N_v) \quad (2)$$

Figure 4b describes the different steps to establish a one hop pair-wise key $K_{u,v}$ between a new node u and a neighbor v.

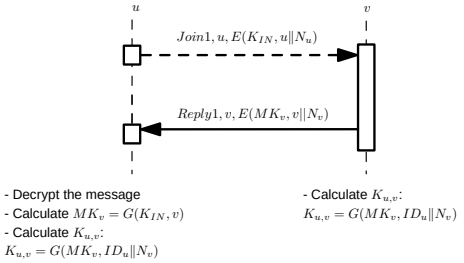
C. Two hops pair-wise key establishment

Establishing "two hops keys" is an important concept of SEDAN. Using two hops keys, SEDAN gets rid of using μTESLA or referring to the base station during the verification process, enabling a scalable secure aggregation solution.

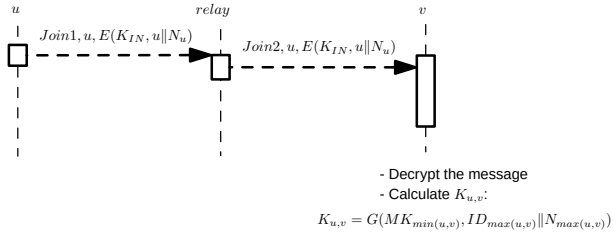
The establishment of the two hops keys is done in parallel with one hop keys. When a new node sends the message



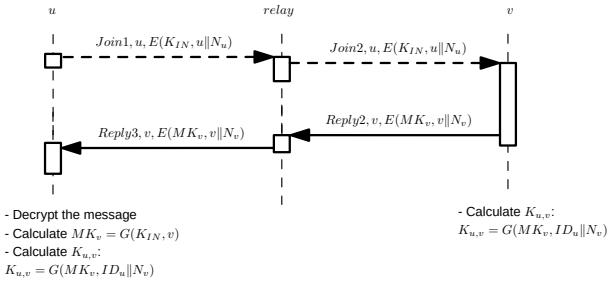
(a) First case of one hop key establishment



(b) Second case of one hop key establishment



(c) First case of two hops key establishment



(d) Second case of two hops key establishment

Fig. 4. EPKE key establishment mechanism between one hop and two hops neighbors. The dashed lines represent broadcast messages.

$Join1$, each receiving neighbor acts as a *relay node* toward the two hops neighborhood. Depending on wether the two hops neighbor erased its transitory initial key or not, we distinguish between two cases:

1) Case 1: As for the first type of keys, a two hops neighbor v still having K_{IN} can compute a pair-wise key with the new node u after receiving the $Join2$ message from the relay node, using the formula (1).

2) Case 2: If the two hops neighbor v erased K_{IN}, the new key must be generated with its master key and its nonce. Thus, v sends the message $Reply2$ to the relay node, which forward it to the new node. The new key is established using the formula (2).

Since the pair-wise key is generated using the master key of u or v, the relay node, if compromised, can not deduce the key, because it does not have K_{IN}.

V. Security Analysis

A. Blind rejection

Blind rejection is an important problem of secure aggregation protocols. A protocol suffering from this kind of problem can not prevent a bogus data from infecting the global aggregation. Our protocol SEDAN overcomes the blind rejection by stopping immediately invalid data during the forwarding phase, before arriving to the sink.

In this section, we illustrate how SEDAN and SAWN react in the presence of malicious nodes. We study the impact of an intruder position on the amount of rejected data in an infected branch[2].

[2]We study only the infected branch because the intruder can not infect other branchs.

We remark that in SAWN and other protocols [3], [11], [4], there is a 100% loss amount. In contrast, SEDAN reacts better: deeper in the tree is the intruder, lesser is the data loss. To simplify the analysis, we consider a binary tree of depth d. We can elaborate the following equation of the data loss DL:

$$DL(x) = \frac{\sum_{k=0}^{d-x} 2^k}{\sum_{k=0}^{d-1} 2^k}, \ 1 \leq x \leq d \tag{3}$$

where x represents the distance of the malicious node from the sink.

Figure 5 illustrates the variation of the data loss over the number of hops between the malicious node and the sink, in a tree of depth 20.

Nevertheless, these results are only applicable when the intruder sends completely false packets. If he forwards correctly the children's data but changes only the aggregation, there is no data loss in SEDAN, because the next hop of the malicious node can reconstitute the real aggregation value.

B. Direct data injection

The direct injection attack occurs when an attacker modifies the data readings reported by the nodes under its direct control [3]. It is very difficult to detect such attacks. In order to reduce their impact, the accepted data reading must be bounded between minimum and maximum values, according to the application.

C. Compromised aggregator node

It is very important to verify the correct behavior of aggregator nodes. To falsify an aggregation value, a compromised node can ignore a received data value or modify it. In the first

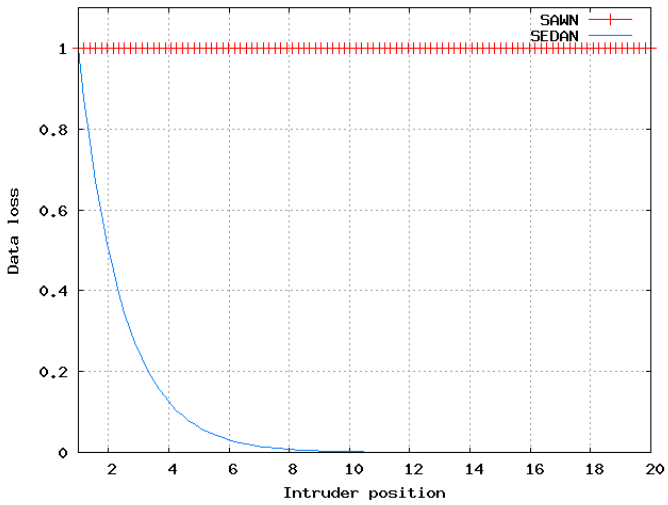


Fig. 5. The amount of rejected data vs. intruder position

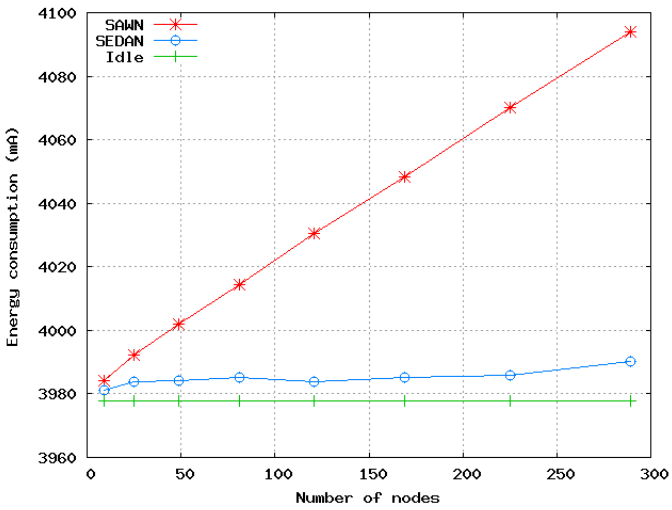


Fig. 7. Energy consumption vs. number of nodes

case, the child can use a simple watchdog mechanism to verify that the parent forwarded its data.

In the second case, SEDAN can block any modification attempt at the next hop level. By verifying the child's OHM and all the grandchildren's THM, a node can detect all possible modifications of the previous aggregator node.

D. Impersonation attack

In SAWN, when a node detects an invalid MAC, it must exclude the two downward nodes (child and grandchild) from the sensor network. However, there is no mechanism in SAWN that enables to verify the origin of a packet. This enables for an intruder to launch an impersonation attack to remove legitimate nodes from the network.

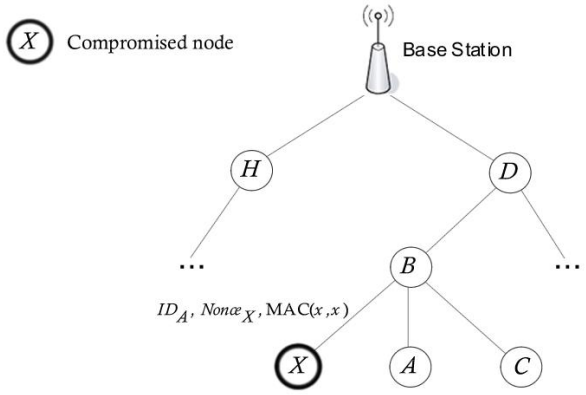


Fig. 6. Impersonation attack

Figure 6 illustrates this attack. The compromised node X may try to send a false message to B using the identity of A. Hence, node D will detect that B or A may be compromised, and will exclude them from the network.

In SEDAN, the use of the pair-wise key between A and B for computing OHM allows data origin authentication, and rejects any message coming from unauthenticated nodes.

VI. Simulations

We have implemented SEDAN and SAWN using the TinyOS [10] environment. All simulations were carried out using the TOSSIM simulator. TOSSIM [16] is a simulator of WSN which compiles a TinyOS application and simulates a network of sensors executing the target application.

For a concise analysis of energy consumption, we have used the PowerTossim [17] extension. This tool gives accurate energy reports based on mica2 motes consumptions: cpu, radio, sensors, ...*etc*.

We have also employed TinySec [18] as a cryptographic library. TinySec contains two cryptographic ciphers: Skipjack and RC5. In our simulations, we have used the Skipjack algorithm for computing encryptions and MACs.

The simulation was run using different grid topologies. Simulation time for all scenarios was fixed to 100 seconds.

In order to show the efficiency of our protocol SEDAN, we have measured two metrics.

A. Energy consumption

Using the PowerTossim extension, we have measured the energy consumption of SAWN and SEDAN.

We have studied the average consumed energy while varying the number of sensors in the network and the number of data packets sent by leaf nodes . To provide a comparison reference, we have also measured the consumed energy by an *idle node*: a node that does not run any protocol.

Figures 7 shows the effect of increasing the number of nodes on the average energy consumption. For these simulations, we have fixed the data rate to 1 packet each 20 seconds. We remark that the power consumption of SEDAN is *very close to the idle mode*. However, SAWN consumes much more energy because it relies on the μTESLA mechanism, witch is not scalable to the large networks.

Figure 8 represents the variation of the energy consumption over the number of transmitted packets. We have fixed the

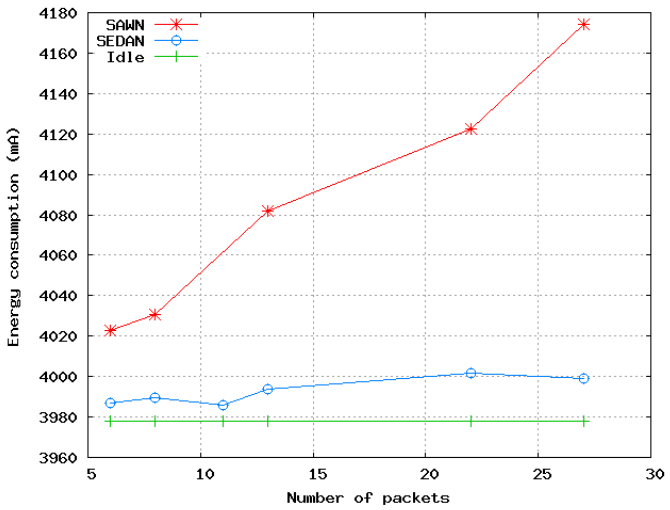


Fig. 8. Energy consumption vs. number of packets

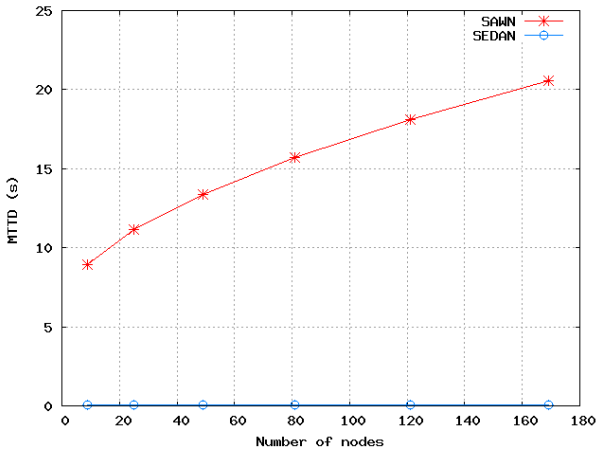


Fig. 9. MTTD vs. number of packets

number of nodes to 81 (a 9x9 grid). When analyzing the protocols behavior by varying the data rate we remark also that SEDAN is more scalable than SAWN, adding only a small overhead comparing to the idle mode.

B. MTTD (Mean Time To Detection)

We mean by "*the mean time to detection* (MTTD)", the average delay between the injection of a bogus packet and its detection.

Figure 9 illustrates the benefit of using the two hops verification mechanism without referring to the base station. Contrary to SAWN, the detection speed in SEDAN is constant and *very close to zero*. Since the detection in SAWN is not distributed, the verification process must be delayed until the completion of the current round and the transmission of all needed keys. This delay increases when the number of nodes increases, since it will require sending more keys (see figure 2).

VII. Conclusion

Saving energy consumption is a compulsory objective in the design of any communication protocol for Wireless Sensor Networks. Indeed, in this kind of networks, sensors are supplied with limited energy batteries, and it is not feasible to replace them after their failure. It is well known that more than 70% of energy is consumed in transmissions in WSN. Therefore, most of this energy can be saved through data aggregation, given that most of the sensed information is redundant due to geographically collocated sensors. However, a second compulsory design objective of any communication protocol for WSN is security. Unfortunately, while aggregation eliminates redundancy (and hence saves energy), it makes data integrity verification more complicated since the received data is unique.

In this paper, we proposed a protocol that guarantees data aggregation while providing efficient data integrity verification mechanisms. Our protocol called SEDAN (Secure and Efficient Data Aggregation protocol for wireless sensor Networks) manages two-hops pairwise keys. This allows to avoid referring to the base station for data integrity verification. Hence, SEDAN minimizes the blind rejection of sensed data. Moreover, SEDAN saves many useless transmissions between sensors and the sink, and thus reduces energy consumption.

We carried out simulations of SEDAN and we compared it to SAWN, using TinyOS environment. The results show that our solution saves energy and minimizes blind rejection while providing the same level of security for aggregated data. Moreover, the mean time to detection (MTTD) in SEDAN is lesser than in SAWN.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks*, pp. 3393 – 422, 2002.

[2] L. Hu and D. Evans, "Secure aggregation for wireless networks," in *Workshop on Security and Assurance in Ad Hoc Networks*, 2003.

[3] H. Chan, A. Perrig, and D. Song, "Secure hierarchical in-network aggregation in sensor networks," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*. New York, NY, USA: ACM Press, 2006, pp. 278–287.

[4] S. Peter, K. Piotrowski, and P. Langendoerfer, "On concealed data aggregation for wireless sensor networks," in *Consumer Communications and Networking Conference, IEEE CCNC 2007*, 2007.

[5] K. Wua, D. Dreefa, B. Sunb, and Y. Xiao, "Secure data aggregation without persistent cryptographic operations in wireless sensor networks," *Ad Hoc Networks*, vol. 5, no. 1, pp. 100 – 111, 2006.

[6] Y. Sang, H. Shen, Y. Inoguchi, Y. Tan, and N. Xiong, "Secure data aggregation in wireless sensor networks: A survey," in *Proceedings of the Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 315–320.

[7] Y. Yang, X. Wang, S. Zhu, and G. Cao, "SDAP: A secure hop-by-hop data aggregation protocol for sensor networks," in *Proceedings of The ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2006.

[8] A. Mahimkar and T.Rappaport, "SecureDAV: A secure data aggregation and verification protocol for sensor networks," in *Proceedings of the IEEE Global Telecommunications Conference*, 2004.

[9] A. Perrig, R. Szewczyk, V. Wen, D. E. Culler, and J. D. Tygar, "SPINS: security protocols for sensor netowrks," in *Proceedings of Mobile Computing and Networking*, 2001, pp. 189–199.

[10] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister, "System architecture directions for networked sensors," in *Proceedings of Architectural Support for Programming Languages and Operating Systems*, 2000, pp. 93 – 104.

[11] M. Raina, S. Ghosh, R. Patro, G. Viswanath, and T. Chadrashekhar, "Secure data aggregation using commitment schemes and quasi commutative functions," *1st International Symposium on Wireless Pervasive Computing*, p. 5, 2006.

[12] S. Zhu, S. Setia, and S. Jajodia, "LEAP: Efficient security mechanisms for large-scale distributed sensor networks," in *Proceedings of ACM CCS*, 2003.

[13] J. Deng, C. Hartung, R. Han, and S. Mishra, "A practical study of transitory master key establishment for wireless sensor networks," in *Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM 05)*, 2005, pp. 289 – 302.

[14] L. Eschenauer and V. D. Gligor, "A key management scheme for distributed sensor networks," in *Proceedings of the 9th ACM Conference on Computer and Communication Security*, November 2002, pp. 41 – 47.

[15] W. Du, J. D. Eng, Y. S. Han, and V. Arshney, "A pairwise key predistribution scheme for wireless sensor networks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS 03)*, 2003, pp. 42 – 51.

[16] L. Philip, L. Nelson, W. Matt, and C. David, "Tossim: Accurate and scalable simulation of entire tinyos applications," in *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*., 2003.

[17] V. Shnayder, M. Hempstead, B. Chen, G. W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*. New York, NY, USA: ACM Press, 2004, pp. 188–200.

[18] C. Karlof, N. Sastry, and D. Wagner, "Tinysec: A link layer security architecture for wireless sensor networks," in *Second ACM Conference on Embedded Networked Sensor Systems (SensSys 2004)*, November 2004.