Efficient data aggregation with in-network integrity control for WSN

Miloud Bagaa, Yacine Challal, Abdelraouf Ouadjaout, Noureddine Lasla, Nadjib Badache

Submitted to JPDC

Submitted to JPDC (2012) 1–17

Efficient data aggregation with in-network integrity control for WSN☆

Miloud Bagaa[a], Yacine Challal[b], Abdelraouf Ouadjaout[a], Noureddine Lasla[a], Nadjib Badache[a]

[a]*Research Center on Scientific and Technical Information (CERIST)*
Algiers, Algeria
Email: {bagaa,aouadjaout,nlasla,badache}@mail.cerist.dz
[b]*Université de Technologie de Compiègne*
Heudiasyc UMR CNRS 6599
PoBox 20529, Compiègne Cedex, France
Email: ychallal@hds.utc.fr

Abstract

Energy is a scarce resource in Wireless Sensor Networks (WSN). Some studies show that more than 70% of energy is consumed in data transmission in WSN. 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 and reduces collisions due to interferences. 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 control integrity for aggregation in 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 numerical analysis and simulations using TinyOS environment. Results show that the proposed protocol yields significant savings in energy consumption while preserving data integrity, and outperforms comparable solutions with respect to some important performance criteria.

Keywords: wireless sensor networks, data aggregation, integrity control.

1. Introduction

In the 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, precision agriculture, fleet management, surveillance and forest fire monitoring.

Node's primary function is to gather measurements 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 energy consumption overhead and collisions. To overcome these problems, many researches have been focused on *aggregating sensed data*, using different mathematical functions: such as MAX, MIN, SUM, AVG,... *etc*. Therefore, only useful aggregation result is sent instead of sending many raw sensed data.

Data aggregation is an essential paradigm for pro-

longing the lifetime of WSN. Indeed, data aggregation reduces the number of broadcasts and hence collisions and energy consumption. However, data aggregation is potentially vulnerable to attackers who may inject bogus information or forge aggregated values without being detected.

At the aggregation module, 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* data integrity control 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 for the overall verification mechanism [6] [7].

The main drawbacks of existing solutions are centralizing the verification process and/or 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 efficient aggregation protocol with in-network integrity verification solving the above problems. Our work is based on two hops verification mechanism: 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 definition and management of a new type of cryptographic keys called *two hops pair-wise keys*. This new type of 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 contributions of our work are many folds:

- First of all, we have reviewed main solutions in the literature and proposed a classification with respect to the communication architecture and the underlying cryptographic scheme used to provide security to data aggregation in WSN;
- Then we proposed, and evaluated through experimentation, a new deterministic and efficient key management scheme (EPKE) which is the cornerstone of our data aggregation protocol with integrity control;
- We have proposed a new efficient aggregation protocol with data integrity control for WSN, called SEDAN;
- In addition to analysis and comparison with respect to important performance criteria, we have defined two new performance criteria specific to data aggregation in WSN. The first one, that we called *blind rejection*, measures the phenomenon of rejecting a global aggregation network view due to few bogus data, despite of important energy consumed in transporting that aggregated data. The second one, that we called MTTD for *Mean Time To Detection* evaluates the delay between data alteration and the detection of this bogus data: lesser is the MTTD, better is the aggregation scheme;
- Finally, we carried out extensive simulations of our protocol SEDAN and other protocols in the literature using TOSSIM/TinyOS augmented with PowerTossim plugin for energy consumption evaluation;

The rest of this paper is organized as follows. Related works are presented in section 2.The design goals of SEDAN and the protocol details are presented in section 3. In section 4, we describe the Extended Pairwise Key Establishment protocol, for establishing all required keys material. In section 5, we provide a security and scalability analysis of our protocol SEDAN. Moreover, we carried out extensive simulations of our protocol under TinyOS [8], and simulation results are detailed in section 6. We draw conclusions in section 7.

2. Related Works

Data aggregation is a collection of data readings that represents a collaborative view of a set of nodes. Also, it aims at increasing the energy saving by reducing the forwarding load of intermediate nodes. When data aggregation is used, intermediate nodes merge multiple packets into one to reduce the amount of transmitted packets. By reducing the number of transmitted messages,

data aggregation also contributes to reducing the number of collisions especially in dense networks. 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 from falsifying the real collaborative view. The aims of securing data aggregation can be summarized in the following points.

- Ensuring authentication and data integrity of aggregation results: data aggregation requires that each intermediate node be able to read and modify the data transmitted in packets. Under this condition, ensuring data integrity becomes challenging. As using a shared key between each node and the sink to secure the data contents denies the use of aggregation mechanisms, nodes use neighborhood pair-wise keying instead. A pair-wise key is a common key shared between a node and its upstream node. It allows the upstream node to manipulate the received data without any control. Thereby, securing data aggregation should allow detecting corrupt aggregation operations, and/or injection of faulty data in the aggregation process.

- Preserving the benefits of data aggregation in terms of energy consumption: the main objective of data aggregation is to reduce energy consumption through minimizing data transmission. Thus, securing data aggregation should not thwart this objective through introducing supplementary computations and transmissions.

2.1. Existing solutions

When an intruder node has no access to the key material of a legitimate node, all the proposed protocols can prevent data aggregation corruption. However, when an intruder obtains the key material of a legitimate node, it can carry out attacks by injecting faulty data or by falsifying the aggregation content. To avoid these problems, many protocols have been proposed in the literature. We propose to classify data aggregation protocols for WSN into two categories, depending on the communication architecture: distributed approaches where aggregation is carried out inside a routing spanning tree, and centralized approaches where aggregation is carried out inside clusters. Furthermore, we refine this classification into three categories, depending on the pairs establishing security associations for integrity control of aggregated data: end-to-end security association, hop-by-hop security association, and hybrid security association, as shown in figure 1.

2.1.1. Protocols based on end-to-end security association

Protocols of this category make use of a shared key between each node and the sink, rooted at the aggregation tree, to guarantee the integrity of the transmitted data. As the data content is encrypted, intermediate nodes use a particular encryption transformation called Privacy Homomorphism (PH) [4] to be able to perform aggregation without disclosing the content of data. The single point of verification in this type of protocols is the sink node that holds all the keys used to encrypt data in the network. This idea has been used in many protocols with different cryptographic techniques such as modular addition that uses a temporary symmetric key CMT[9], or additive property of the complex numbers ASAP [10].

2.1.2. Protocols based on hop-by-hop security association

In contrast to protocols based on end-to-end encrypted data, protocols based on hop-by-hop encryption make use of other mechanisms to guarantee integrity while allowing data aggregation in plain text. To ensure the integrity of data transmitted between nodes, each protocol uses a different verification mechanism. Wu et al [5] proposed a protocol based on watch dog scheme that we call AWPC: Nodes in a same clique can listen to each others, and hence verify the aggregator behavior. Using the Beta reputation system and based on [5] the authors in [11] proposes a new solution which aims to detect malicious nodes that try to falsify data, or do not participate to routing data. Moreover, authors propose in [11] to use a local recovery when malicious nodes are detected, in order to guarantee a service continuity despite the presence of intruders and avoid long term isolation of legitimate nodes.

Based on a cluster architecture authors in [12] propose a new solution, called RSDA, which uses a reputation system to protect the sensing, forwarding and aggregating of data against the malicious nodes. Using a watchdog technique, a reputation system and a voting mechanism, the malicious nodes, in each cluster, can be detected and then black listed.

In [3], Chan et al. proposed a protocol based on the concept of commitment tree, that we call SHAN. The verification of aggregation values takes place at the leaf nodes level. The base station sends a proof of the final aggregation to leaf nodes. Each leaf node reconstitutes

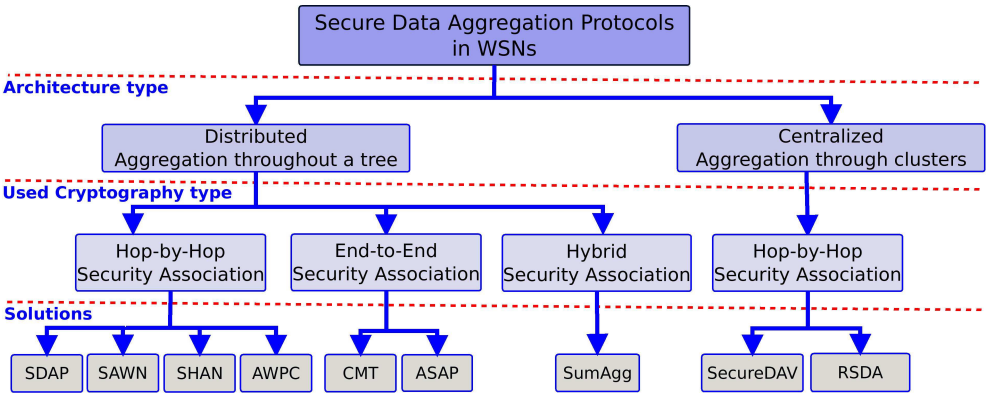


Figure 1: Classification of Secure Data Aggregation Protocols

the intermediate aggregation values of each hop toward the sink. The final result is compared to the sink's proof.

Yang et al [13] propose a new solution which uses a *divide-and-conquer* approach called SDAP. 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. The verification of the aggregation value holds then at the base station.

Mahimkar and Rappaport propose a protocol in [14], called SecureDAV, 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 [15]. The sink verifies then the signatures before considering the aggregation value.

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 detailed description of SAWN and analyze its main drawbacks.

2.1.3. Protocols based on hybrid security association

A new hybrid solution called SumAgg is proposed in [16] which aims to provide both flexible data aggregation like in hop-by-hop approach and optimal data confidentiality like in end-to-end protocols. Using a pair-wise key shared between each node and its parent, each leaf node in the tree sends through a secure way to its parent two values f_1 and f_2. The latter, are the encryption of the sensed data with a Privacy Homomorphism (PH) [4] using two different pair-wise keys shared with the base station. After the reception of all children messages by the parent node, the latter uses the PH function to compute the aggregation of $f_1 - child$ values and the aggregation of $f_2 - child$ values without need for disclosing their content. This way, the sink receives two encrypted aggregation values f_1 and f_2. Then, the sink decrypts the two received values using the secret keys of the nodes participating to the aggregation and hence the calculation of the two values f_1 and f_2. The verification of the integrity of the aggregation result is performed by comparing the decryption results of f_1 and f_2. The result of the aggregation is accepted in the case of equality between the two decryption results, and in this case the aggregation value is the decryption of either f_1 or f_2. Otherwise, by exploiting the hop-by-hop security associations and a secure broadcast authentication protocol, such as μTESLA [17], a commitment and attestation phase will be launched to localize the compromised node in the network.

2.2. Terminology and Notation

In table 1 we introduce the different notations and terminology used throughout this paper to describe the different solutions and our own protocols.

2.3. 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 simultaneously. In addition to the two hops verification mechanism, SAWN is based on the concept of *delayed verification*, which leads to some important drawbacks. Figure 2 explains the aggregation process in SAWN. At each round r, each leaf node of the tree sends its measurement reading and a Message

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 1: Notations

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:

$$\begin{aligned} C \rightarrow D : \quad & ID_A, d_A, MAC(K_A, d_A) \\ & ID_B, d_B, MAC(K_B, d_B) \\ & MAC(K_C, f(d_A, d_B)) \end{aligned}$$

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.

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

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 protocol [17].

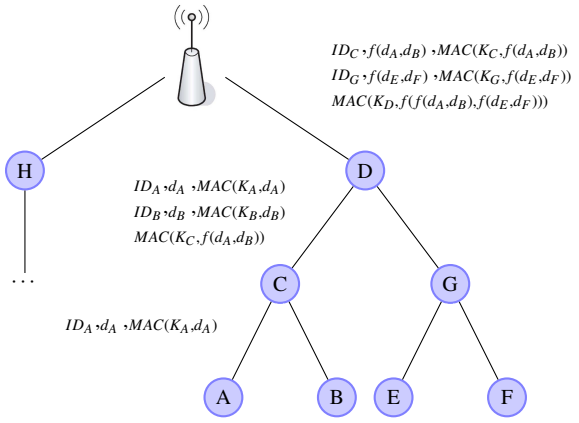


Figure 2: Aggregation process in SAWN

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 [8], the default packet size is 29 bytes. So, each packet can transmit only the revelation of at most 4 nodes.

Figure 3 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, 18, 4, 14], the violation of data integrity at any place in the network obligates the sink to reject the received aggregation data. Since the latter represents the view of the whole *infected branch*[1], this rejection yields a significant data loss.

Moreover, since nodes are responsible for data veri-

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

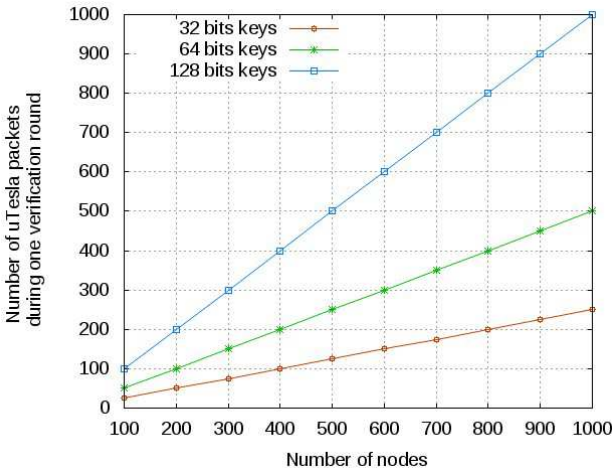


Figure 3: Number of required μTesla packets during one verification round vs. number of nodes

fication 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.

3. Our Solution: SEDAN

3.1. Design Goals

To overcome the above constraints, our solution overcomes the need for a whole broadcast in the network through 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 tampered data at the compromised node level, avoiding transmitting it to the sink and infecting other correct aggregation values. Hence, all aggregation results arriving at the base station are correct and can be committed *immediately*.

3.2. Assumptions

- The aggregation function should verify the following property [18]:

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

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 [2] assumption: the tree should not contain two consecutive compromised nodes.

3.3. Description

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

3.3.1. Key establishment

The initial step consists of establishing all needed pair-wise keys. Note that this phase 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.

3.3.2. Data authentication

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

$$ID_i, N_i, d_i,$$
$$MAC(K_{i,P(i)}, N_i \parallel d_i),$$
$$MAC(K_{i,GP(i)}, d_i \parallel N_i)$$

The nonce N_i prevents 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 the 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.

3.3.3. 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}.

3.3.4. 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 :

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

and its THM :

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

Then, node j transmits to its parent :

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

3.3.5. 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 relay is 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 reiterates the step 4 by sending:

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

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 $OHM = MAC(K_{j,P(j)}, N_j \parallel d_j)$. Node j sends also its THM on the aggregation value over its reading and children's data.

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

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 4.

3.4. Example

Figure 3.4 illustrates the messages exchanged during an aggregation using our protocol SEDAN.

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

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

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

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

- 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.

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

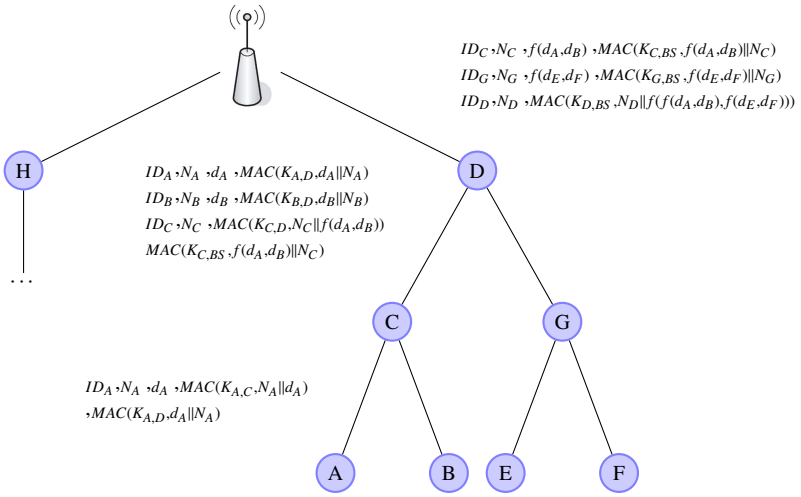


Figure 4: Aggregation process in SEDAN

4. EPKE : Extended Pair-wise Key Establishment protocol

There exists considerable work [19, 20, 21, 22, 23, 24] dealing with the problem of securing communication infrastructure within a network of sensor nodes. Given the deployment nature of WSN and their resource constraints, the most appropriate scheme would be ideally self-organized without prior knowledge of deployment and based on a symmetric key management mechanism.

Indeed, 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 [21] 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 [19, 20] 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. This short interval of time is so short that it would be impossible for an intruder to steal a sensor node, to connect it to a terminal and to do the required steps to dump and analyse its memory to find out the cryptographic material.

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 [19] and OTMK [20].

4.1. Initialization

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

4.2. 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:

4.2.1. 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:

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

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 5a describes the different steps to establish a one hop pair-wise key $K_{u,v}$ between u and a neighbor v within $T_{\min}$.

4.2.2. Case 2

After $T_{\min}$, previously deployed nodes would have erased 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) :

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

Figure 5b describes the different steps to establish a one hop pair-wise key $K_{u,v}$ between a new node u and a neighbor v.

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

4.3.1. 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).

4.3.2. 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 forwards it to the new node. The new key is established using the formula (2).

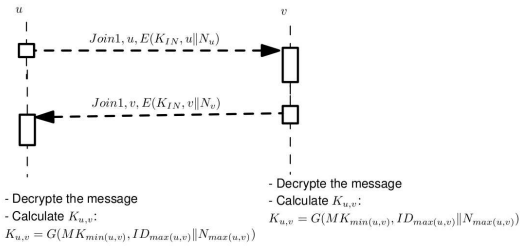
Since the pair-wise key is generated using the master key of u or v, the relay node, if compromised, can not deduce the key, because it does not have K_{IN}.

4.4. Tmin determination through experimentation

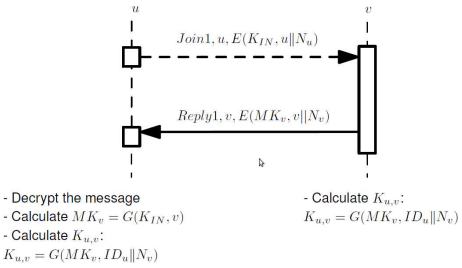
Our protocol SEDAN relies, like LEAP [19] and RSDA [12] and may be others, on the assumption that an intruder cannot have access to the cryptographic material during the initialization phase Tmin that is too short. Indeed, an intruder requires a minimum of time to take the victim node, to connect it to a serial port, to dump its memory and analyze its content for any cryptographic material such as the initialization key K_{IN}. We were interested in determining the value of Tmin that allows establishing the two hops session keys and then erasing the initial key K_{IN}. For this end, we have implemented our key management protocol EPKE over a WSN platform composed of MicaZ nodes of Memsic [25]. The considered topology is depicted in figure 6. We carried out extensive tests, each time we consider a value of Tmin and calculate the percentage of succeeded one hop and two hops session keys establishment. The results are plotted in figure 7. We notice that after three seconds, EPKE succeeds to establish most of the required session keys: 100% of the one hop session keys, and 100% of the two hops session keys. We notice also that few nodes (negligeable) miss to establish the two hops session keys at all because of packet losses due to interference. According to these results, we believe that Tmin=3s is a good value that allows to EPKE to finish the session keys establishment, and does not suffice for an intruder to carry out the necessary steps to compromise a mote and get out the cryptographic material.

Moreover, this minimal confidence time interval does not depend on the network size. Indeed, our protocol EPKE relies exclusively on local interactions through broadcast messages. Thus, the network size does not effect the required time to establish the two hops pair-wise keys.

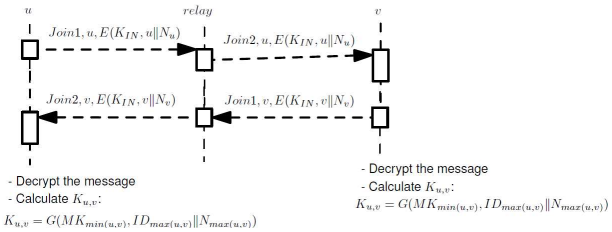
In order to verify this hypothesis, we carried out extensive simulations using TinyOS/TOSSIM simulator [26]. We considered random topologies with average density equal to the experimental configuration (each node has 5 neighbors in average), and increased the network size up to 600 nodes. For each scenario, we have run the simulations 35 times and calculated the 95% confidence interval. The results are plotted in figure 8 where the confidence intervals are plotted as vertical bars. This graph confims our above hypothesis and



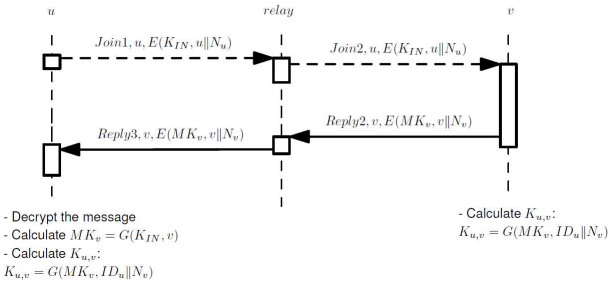
(a) First case of one hop key establishment



(b) Second case of one hop key establishment



(c) First case of two hops key establishment



(d) Second case of two hops key establishment

Figure 5: EPKE key establishment mechanism between one hop and two hops neighbors. The dashed lines represent broadcast messages.

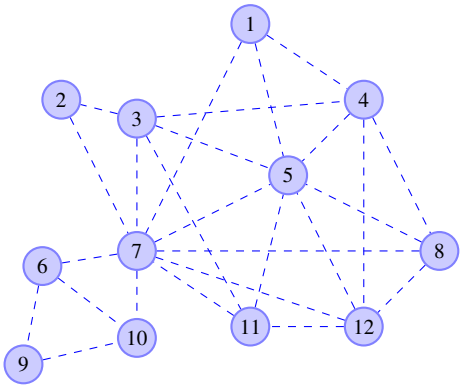


Figure 6: The network toplogy of Micaz motes

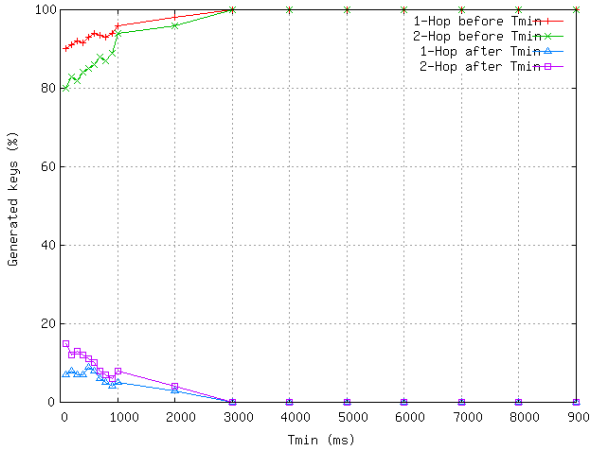


Figure 7: The number of generated keys vs. Tmin

demonstrates that EPKE scales well to large networks because of relying exclusively on local broadcast interactions.

5. Analysis

In this section, we provide security and scalability analysis of our protocol SEDAN compared to some representative solutions in the literature.

5.1. Security analysis

5.1.1. 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 reacts in the

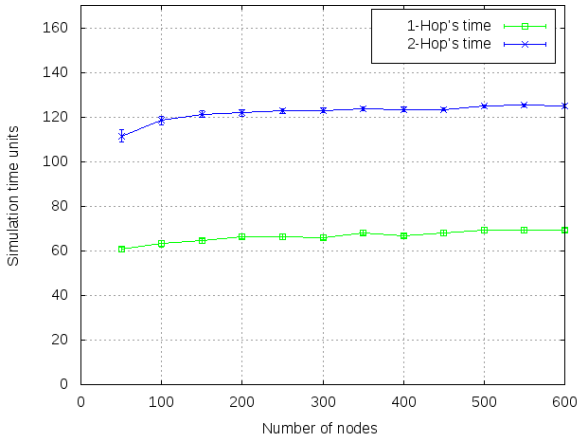


Figure 8: Impact of network size on Tmin

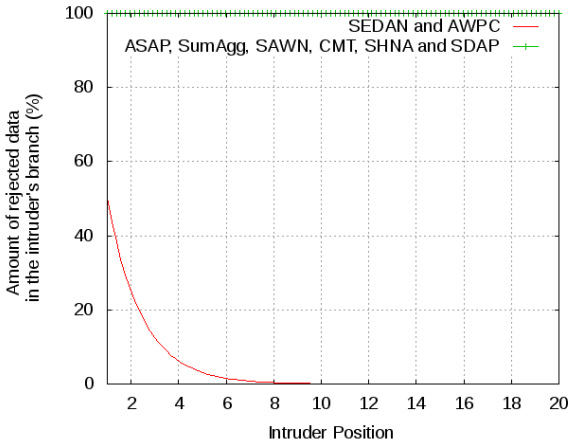


Figure 9: The amount of rejected data vs. intruder position

presence of malicious nodes. We study the impact of an intruder position on the amount of rejected data in an infected branch[2].

To simplify the analysis, we consider a binary tree of depth d. We can elaborate the following equation to calculate the number of lost packets DL (data loss):

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

where x represents the distance of the malicious node from the sink. Figure 9 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. SEDAN and AWPC [5], by stopping immediately invalid data during the aggregation process, overcome the blind rejection of the final aggregation value. However, in SAWN[2], CMT[9], ASAP [10], SumAgg[16], SDAP[13] (assuming there is a single leader), SHAN[3] and due to the fact that the verification is done at the sink level, the final aggregation value is rejected after it has been relayed up to the sink.

Moreover, 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.

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

5.1.2. *Resilience against aggregator node capture*

In any aggregation protocol, it is very important to verify the behaviour of aggregator nodes. A compromised aggregator node can falsify the aggregation value by rejecting the received data value from its children or simply modifying it. In the first case, all protocols, except CMT, prevent such attacks by using a simple watchdog mechanism. In the second case, SAWN under the assumption that two consecutive nodes cannot be compromised and by employing the two hops verification mechanism, detect any modification tentative at the parent node level. AWPC by using a watchdog mechanism in the same clique can detect this kind of attacks if the number of compromised nodes is smaller than n/2, with n being the number of nodes in a given clique. SEDAN can block any modification attempt at the next hop level. By verifying the childs OHM and all the grandchildrens THM, a node can detect all possible modifications of the previous aggregator node. In SecureDAV, the aggregation values sent by each cluster head are verified by the sink node. Each cluster head sends, in addition to the mean of the received data, the mean data signatures of some of its cluster members. However, detecting the cluster head compromise cannot be guaranteed if some of the cluster members are also compromised. CMT uses the PH encryption in the purpose of detecting any faulty aggregation value. However, authors in [4] show that it is possible for an attacker to alter the encrypted aggregation value without knowledge of the plaintext, which forbids the detection of an existing compromised aggregator node.

5.1.3. 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 semantically checked according to the application.

5.1.4. Impersonation attack

Impersonation attack is the possibility of launching an attack by injecting false data carrying the source address of another node. If this attack happens, the pretended source will be considered as an intruder and then, will be revoked from the network. In SDAP, SumAgg, SEDAN, RSDA and SecureDAV, the use of the pairwise key between a node and its upstream allows data origin authentication, and rejects any message coming from unauthenticated nodes. All end-to-end encryption protocols except ASAP miss a local authentication mechanism, allowing the intruder to execute an impersonation attack. The AWPC protocol is vulnerable to the impersonation attack. Indeed, a malicious node can send a faulty data to one selected parent, using the identity of one chosen member node belonging to the same clique of the selected parent. To lunch such an attack, the malicious node must be positioned in the neighborhood of the parent. In this case, the parent calculates a fault aggregation value and hence will be considered as a malicious node by its child members. In SAWN and similar solutions, when a node detects an invalid MAC, it must exclude the two downward nodes (child and grandchild) from the sensor network. Indeed, there is no mechanism in SAWN that allows verifying the origin of a packet. This enables for an intruder to launch an impersonation attack to remove legitimate nodes from the network.

Figure 10 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.

5.1.5. Loclaization

When detecting faulty data aggregation values, it is important to drop them, and also localize the compromised node to revoke it from the network. The protocols based on end-to-end encrypted data suffer from the lack of localization of the intruder, since the sink receives and verifies only the final result. However, the localization of malicious nodes in the protocols based on hop-by-hop encryption (such as our solution SEDAN) is possible because intermediate nodes have access to payload data and thus can detect the malicious nodes that falsify the aggregation.

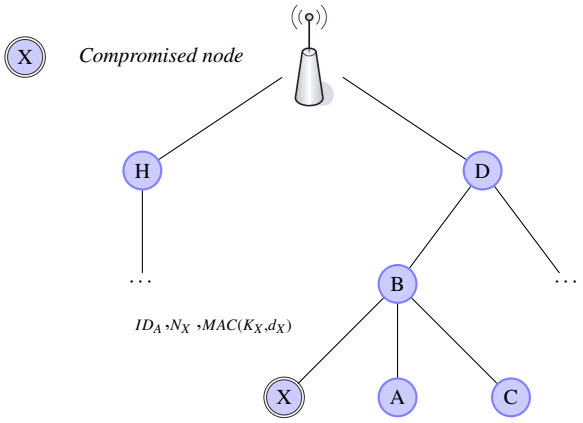


Figure 10: Impersonation Attack

5.2. Scalability analysis

5.2.1. Impact of network size

Depending on the mechanism used to secure data aggregation, protocols react differently when the network size increases. The revelation of keys in SAWN and SumAgg are based on the assumption that the sink node can reach all sensor nodes in the network, using only one hop broadcast. When the network size increases, this assumption will be hardly verified. Therefore, SAWN and SumAgg do not scale well with large networks. SecureDAV, which assumes that cluster heads send the aggregation value through only one hop to the sink node, doesn't satisfy also scalability requirements. Protocols SEDAN, RSDA, SHAN, AWPC and SDAP, however, are scalable because they rely on a distributed verification mechanism and do not make any reference to the sink node. Furthermore, CMT and ASAP that are similar to a simple aggregation process offer a better scalability too.

5.2.2. The storage and bandwidth overheads

Typical WSN nodes suffer from resource constaintes, especially energy, bandwidth and storage limitations. We will evaluate the energy overhead in section 6 through simulations, and in what follows, we will measure storage and bandwidth overheads. We will consider specifically grand-parent's overheads that are maximal compared to the other nodes of the network, given the typical interactions of in-tree agregation protocols. We assume that a grand-parent node P_i has C_i children and G_i grand children as illustrated in figure 11. Later we

generalize the results for a simplified topology, namely a balanced complete tree.

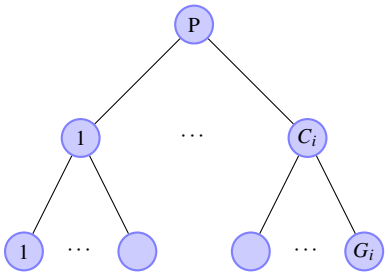


Figure 11: Considered topology for scalabilty analysis

The storage overhead. The verification in SEDAN is instantaneous and hence there is no need for saving the received messages. However in SAWN, before the phase of nodes' keys disclosure, each node it must save in its local memory the whole received messages from their children and grandchildren. Therefore, the storage overhead ($|D_{Saved}|$) can be evaluated as follows:

$$|D_{Saved}| = Ci \times (|MAC|) + Gi \times (|ID + Data + MAC|)$$

To perform numerical calculation let us assume the MAC size to be four bytes and both ID and data sizes be two bytes. Thus, the size of saved data becomes

$$|D_{Saved}| = 4 \times Ci + 8 \times Gi \ \ bytes$$

Assuming our aggregation structure is a balanced complete tree, $G_i = C_i^2$, figure 12 illustrates the storage overhead in both SEDAN and SAWN, compared to a reference point which is the RAM size of Micaz. We can notice that in the case of SAWN, starting from a threshold of children nodes size, the memory is saturated with the saved data without taking into consideration the size of the program code, keys materials and other data.

Bandwidth overhead. Bandwidth is a scarce resource in WSN in order to minimize energy consumption. In order to give an idea about the induced bandwidth overhead we will evaluate in what follows the size of sent and received data induced by our protocol exchanges SEDAN and SAWN and make a comparison.

In SEDAN and SAWN a grand parent node receives data from its children and grand children, to make verifications and then calculate its aggregation and transmit it upward. We assume that the size of a key is four bytes, and the size of a nonce is one byte. Then, the size of

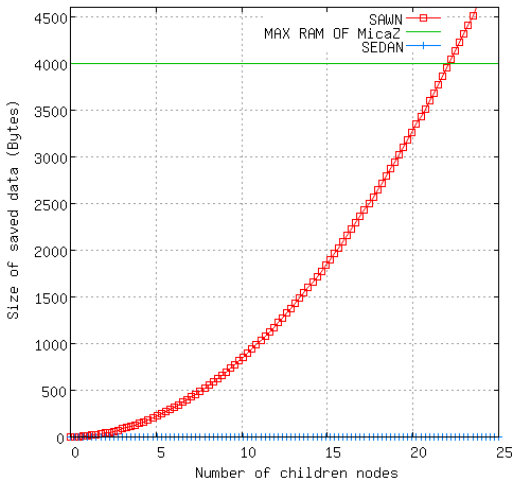


Figure 12: Size of saved data (bytes) vs. number of children nodes

received data ($|D_{Received}|$) in the case of SEDAN can be evaluated as follows:

$$\begin{aligned} |D_{Received}| &= G_i \times |ID + Nonce + Data + MAC| \\ &\quad + C_i \times |ID + Nonce + MAC + MAC| \\ &= 9 \times G_i + 11 \times C_i \ \ bytes \end{aligned}$$

and the amount of sent data ($|D_{Sent}|$) is

$$\begin{aligned} |D_{Sent}| &= C_i \times |ID + Nonce + Data + MAC| \\ &\quad + |ID + Nonce + MAC + MAC|) \\ &= 9 \times C_i + 11 \ \ bytes. \end{aligned}$$

In the case of SAWN, the size of received data is

$$\begin{aligned} |D_{Received}| &= G_i \times |ID + Data + MAC| \\ &\quad + C_i \times |MAC| \\ &= 8 \times G_i + 4 \times C_i \ \ bytes. \end{aligned}$$

and the amount of sent data is

$$\begin{aligned} |D_{Sent}| &= C_i \times |ID + Data + MAC| + |MAC| \\ &= 8 \times C_i + 4 \ \ bytes. \end{aligned}$$

and also the amount of received keys ($|K_{Received}|$) from the sink in order to verify the data integrity by the grand parent :

$$\begin{aligned} |K_{Received}| &= (G_i + C_i) \times size(key) \\ &= 4 \times (G_i + C_i) \ \ bytes. \end{aligned}$$

So the number of total received messages ($|M_{Received}|$) is :

$$\begin{aligned} |M_{Received}| &= |D_{Received}| + |K_{Received}| \\ &= 12 \times G_i + 8 \times C_i \ \ bytes. \end{aligned}$$

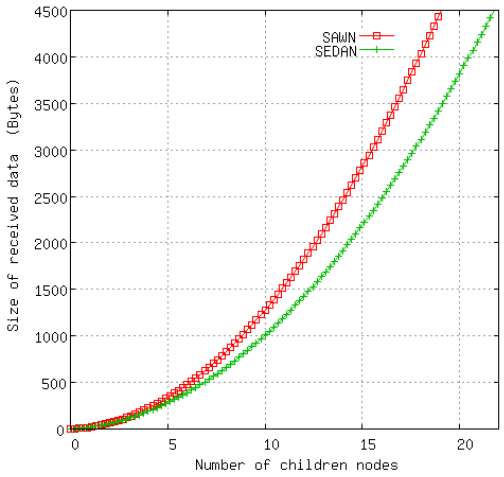


Figure 13: Size of received data (bytes) vs. number of children nodes

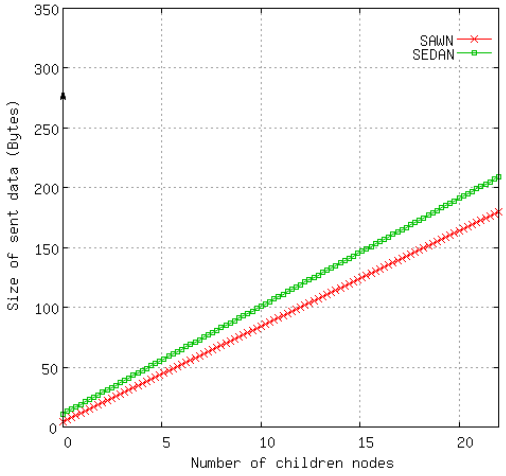


Figure 14: Size of sent data (bytes) vs. number of children nodes

Figure 13 shows the impact of varying the number of children on the number of received messages in SEDAN and SAWN in the case of a balanced tree. Figure 14 shows the impact of varying the number of children on the number of sent messages.

6. Simulation Results

We have implemented SAWN, CMT, SecureDAV, SDAP, SHAN and SEDAN using the TinyOS [8] environment. All simulations were carried out using the TOSSIM simulator. TOSSIM [26] is a simulator of WSN which compiles a TinyOS application and simulates a network of sensors executing the target application.

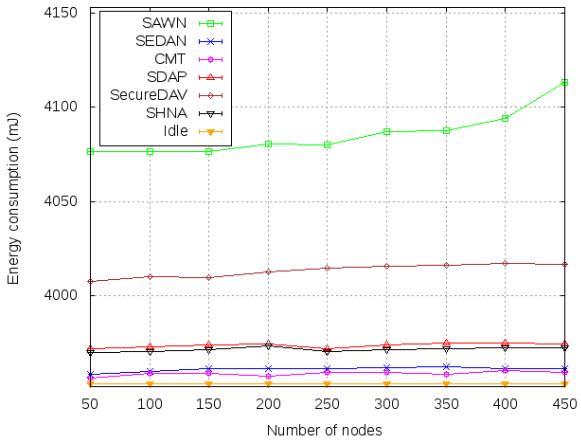


Figure 15: Energy consumption vs. number of nodes

For a concise analysis of energy consumption, we have used the PowerTossim [27] extension. This tool gives accurate energy reports based on mica2 motes consumptions: cpu, radio, sensors, ... *etc*.

We have also employed TinySec [28] 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 random topologies with average density equal to the experimental configuration in section 4 (each node has 5 neighbors in average). We increased the network size up to 450 nodes. 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: energy consumption, and the mean time to bogus data detection (MTTD).

6.1. Energy consumption

Using the PowerTossim extension, 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 a *idle node*: a node that does not run any protocol.

Figure 15 illustrates the consumed energy by each protocol while increasing the size of the network up to 450 sensors. We remark that the energy consumption depends on the verification mechanism used by each protocol. For example in SAWN the broadcast used by the sink node to reveal the keys used in MAC computation and verification to the entire network, consumes an important amount of energy. SEDAN relies

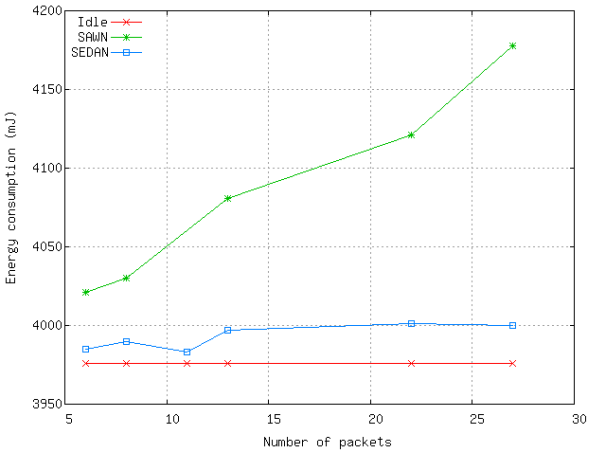


Figure 16: Energy consumption vs. number of packets

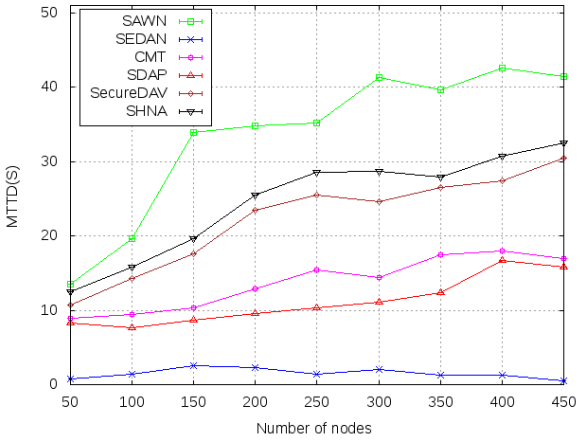


Figure 17: MTTD vs. number of nodes

also on MAC computation to guarantee data authentication and integrity. However, we remark that the power consumption of SEDAN is very close to the *idle mode*. SEDAN outperforms SAWN because it reduces key transmissions. Indeed, in SEDAN each node shares MAC keys with its parent and grandparent through local exchanges. Whereas, in SAWN, the sink node broadcasts periodically, to the entire network, the key material to be used in MAC computation. The use of ECC in SecureDAV to sign the average data consumes a non negligible amount of energy. SDAP induces low energy consumption overhead thanks to the divide-and-conquer approach where leader nodes send aggregated data to the sink. SHAN protocol induces also a low energy overhead. This is due to the verification processes that is limited to local verifications at the leaf and sink nodes. CMT consumes a negligible amount of energy that is similar to a simple aggregation process.

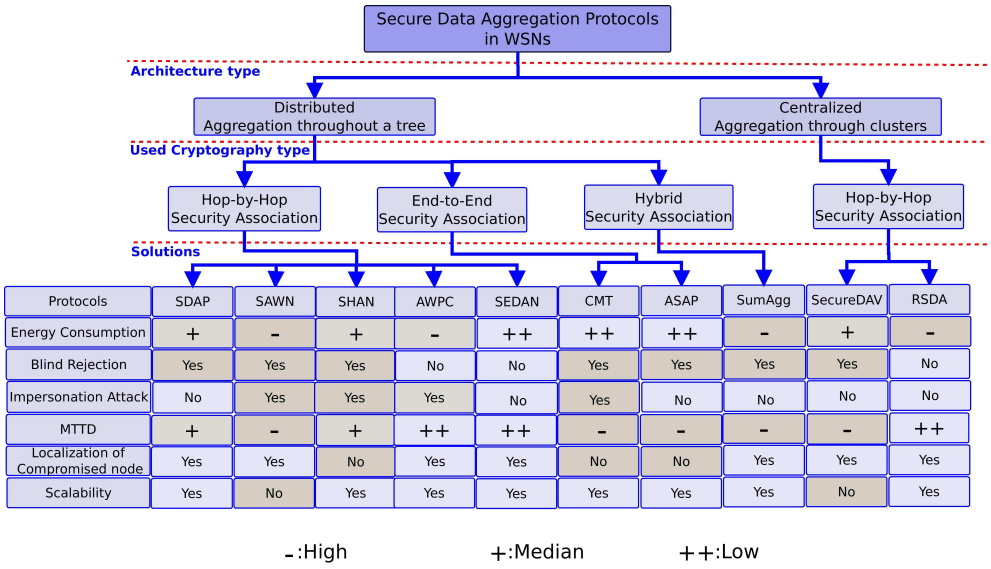
Figure 16 represents the variation of the energy consumption over the number of transmitted packets. We have fixed the number of nodes to 81 (a 9x9 grid). When analyzing the protocols behavior by varying the data rate we remark also that SEDAN is more scalable, adding only a small overhead comparing to the idle mode.

We notice that there is a tradeoff between energy consumption and intruder localization: as we can see in [4], while CMT and ASAP (which are an end-to-end encryption protocols) suffer from the lack of localization of the intruder, they do not consume a lot of energy. However, SecureDAV and SAWN do not suffer from the lack of localization of the intruder but introduce extra transmission in order to coordinate incorrect aggregations whenever they appear. Our solution SEDAN overcomes this trade-off by localizing intruder nodes with minimum number of transmitted messages.

6.2. *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 17 illustrates the mean time needed for each protocol, before the bogus injected data can be detected. SEDAN uses a totally in-network verification mechanism that makes the detection speed, constant and very close to zero. However, since the detection in SAWN, SDAP, CMT, SHAN and SecureDAV are not distributed, the verification process must be delayed until the reception of all data by the sink node. Moreover, SAWN needs additional time to reveal all required keys to the whole network. This time increases when the number of nodes increases, since it will require sending more keys (see figure 3). Similarly, in SHAN leaf nodes wait for the reception of the proof value calculated at the sink before committing the validation of the aggregation value. This supplementary step increases the time to detection of bogus data. The use of ECC that requires more execution time than the symmetric cryptography, interprets the low detection speed in SecureDAV over CMT.

In figure 18, we summarize the analysis and the performance evaluation presented in sections 5 and 6. We notice that our solution SEDAN provides good tradeoffs with respect to the considered security and performance criteria compared to the considered sample of secure aggregation protocols for WSN.



Protocols	SDAP	SAWN	SHAN	AWPC	SEDAN	CMT	ASAP	SumAgg	SecureDAV	RSDA
Energy Consumption	+	-	+	-	++	++	++	-	+	-
Blind Rejection	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	No
Impersonation Attack	No	Yes	Yes	Yes	No	Yes	No	No	No	No
MTTD	+	-	+	++	++	-	-	-	-	++
Localization of Compromised node	Yes	Yes	No	Yes	Yes	No	No	Yes	Yes	Yes
Scalability	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes

Figure 18: Taxonomy of Secure Data Aggregation Protocols

7. Conclusion

Reducing 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 and comparisons of CMT, SAWN, SecureDAV, SDAP, SHAN and SEDAN, 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 bogus data detection (MTTD) in SEDAN is lesser than other solutions, and our solution outperforms other representative solutions of the litterature with respect to some important performance criteria.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: A survey, Computer Networks (2002) 393 – 422.

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

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

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

[5] K. Wu, D. Dreefa, B. Sunb, Y. Xiao, Secure data aggregation without persistent cryptographic operations in wireless sensor networks, Ad Hoc Networks 5 (1) (2006) 100 – 111.

[6] A. Ouadjaout, M. Bagaa, A. Bachir, Y. Challal, N. Lasla, L. Khelladi, Information Security in Wireless Sensor Networks, World Scientific, 2009, pp. 427–472.

[7] Y. Sang, H. Shen, Y. Inoguchi, Y. Tan, 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, IEEE Computer Society, Washington, DC, USA, 2006, pp. 315–320.

[8] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, 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.

[9] C. Castelluccia, E. Mykletun, G. Tsudik, Efficient aggregation of encrypted data in wireless sensor networks, in: Mobile and Ubiquitous Systems: Networking and Services, INRIA, Cedex, France, 2005, pp. 109–117.

[10] R. Bista, K.-J. Jo, J.-W. Chang, A new approach to secure aggregation of private data in wireless sensor networks, in: Dependable, Autonomic and Secure Computing, 2009. DASC '09. Eighth IEEE International Conference on, 2009, pp. 394 – 399.

[11] H. Feng, G. Li, G. Wang, Efficient secure in-network data aggregation in wireless sensor networks, Networks Security Wireless Communications and Trusted Computing (NSWCTC) 1 (2010) 194 – 197.

[12] H. Alzaid, E. Foo, J. Nieto, Rsda: Reputation-based secure data aggregation in wireless sensor networks, in: International Conference on Parallel and Distributed Computing, Applications and Technologies, 2008, pp. 419 – 424.

[13] Y. Yang, X. Wang, S. Zhu, 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.

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

[15] B. Ian, S. Gadiel, S. Nigel, Advances in Elliptic Curve Cryptography, Combridge University Press, 2005.

[16] E. Mlaih, S. Aly, Secure hop-by-hop aggregation of end-to-end concealed data in wireless sensor networks, in: INFOCOM Workshops 2008, IEEE, 2008, pp. 1 – 6.

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

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

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

[20] J. Deng, C. Hartung, R. Han, 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.

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

[22] W. Du, J. D. Eng, Y. S. Han, 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.

[23] W. Bechkit, A. Bouabdallah, Y. Challal, Enhancing resilience of probabilistic key pre-distribution schemes for wsns through hash chaining, in: ACM (Ed.), Proceedings of the 17th ACM conference on Computer and communications security (CCS 2010), 2010, pp. 642–644.

[24] W. Bechkit, Y. Challal, A. Bouabdallah, A. Bencheikh, An efficient and highly resilient key management scheme for wireless sensor networks, in: Proceedings of the 35th IEEE Conference on Local Computer Networks (IEEE LCN 2010), 2010, pp. 220–223.

[25] memsic, http://www.memsic.com.

[26] L. Philip, L. Nelson, W. Matt, 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.

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

[28] C. Karlof, N. Sastry, D. Wagner, TinySec: a link layer security architecture for wireless sensor networks, in: Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SensSys 2004), 2004.