Securing wireless sensor networks against aggregator compromises

Thomas Claveirel¹, Marcelo Dias de Amorim¹, Michel Abdalla², and Yannis Viniotis³

¹ LIP6/CNRS
Université Pierre et Marie Curie – Paris 6
Paris, France
claveiro,amorim}@rp.lip6.fr

² Département d’Informatique
École Normale Supérieure
Paris, France
michel.abdalla@ens.fr

³ Department of ECE
North Carolina State University
Raleigh, NC, USA
candice@ncsu.edu

Abstract—A common approach to overcome the limited nature of sensor networks is to aggregate data at intermediate nodes. A challenging issue in this context is to guarantee end-to-end security mainly because sensor networks are extremely vulnerable to node compromises. We propose three schemes to secure data aggregation that rely on multipath routing. The first guarantees data confidentiality through secret sharing, while the second and third provide data availability through information dispersal. Based on qualitative analysis and implementation, we show that, by applying these schemes, a sensor network can achieve data confidentiality, authenticity, and protection against denial of service attacks even in the presence of multiple compromised nodes.

Index Terms—Wireless sensor networks, data aggregation, node compromise.

I. INTRODUCTION

Radio communications are expensive in terms of energy consumption. In wireless sensor networks (WSN), where nodes might be very limited in resources, it is fundamental that communication overhead be controlled. An interesting approach to achieve such an objective is to perform data aggregation, where relaying nodes exploit the distributed nature of the network and perform in-network processing. This means that multiple readings from various sensors are merged into smaller messages as they are conveyed toward the base station. A common practice, for example, is for intermediate nodes to compute and forward the mean of the readings they received so far instead of the readings themselves.

Guaranteeing security in aggregation schemes is particularly challenging because node compromises in such a scenario are doubly problematic, both in terms data confidentiality (eavesdropping) and availability (denial of service). An attacker that compromises a node has access to its internal state and cryptographic material. It may therefore turn an authorized node into a malicious one. Indeed, by compromising an aggregator node, the attacker would endanger all of the readings that are part of the aggregate the node is in charge of.

In this paper, we focus on confidentiality and availability, which still lack efficient solutions (cf., Section II). We do not address data integrity as an explicit issue. We propose, analyze, and evaluate three new schemes, namely (a) Secret Multipath Aggregation (SMA), (b) Dispersed Multipath Aggregation (DMA), and (c) Authenticated Dispersed Multipath Aggregation (A-DMA). The main idea behind our three approaches is to exploit using multiple paths toward the sink. In fact, a sensor may split a handful of its readings into $n$ separate messages such that $t$ messages are needed to reconstruct the readings. By sending messages along disjoint paths, a sensor ensures that intermediate nodes do not have complete knowledge of the sensed data. Every sensor splits its readings the same way, and intermediate nodes may aggregate messages by adding them and multiplying them by a constant. When the sink recovers $t$ aggregated messages, it may reconstruct an aggregate of the readings. In such a scenario, SMA guarantees confidentiality by applying the concept of secret sharing [1]. DMA and its authenticated version, A-DMA, address availability by dispersing information over the different paths [2]. Although they have been recognized in many research areas (e.g., parallel computing, distributed storage, databases, and ad hoc networking), surprisingly neither secret sharing nor information dispersal have been applied to the context of wireless sensor networks nor to the specific problem of data aggregation.

II. RELATED WORK

Several researchers have already studied the problem of securing data aggregation. Mykletun et al. [3] suggest using ciphers for which some arithmetical operations over ciphertexts have arithmetical significations on the cleartext. While this technique allows for some security, a compromised node may still stop aggregating and forwarding data. Even worse, tampering and replay attacks cannot be detected with such a solution. Przydatek et al. [4] propose a number of commit-and-prove techniques to ensure the integrity of the aggregated data for some aggregation functions. Integrity can be assured (although in a probabilistic fashion), but the proposed schemes are difficult to implement and provide neither confidentiality nor protection against denial of service (DoS) attacks. Yang et al. [5] also propose a similar technique that suffers from the same drawbacks. None of the works ensures confidentiality upon insider attacks. Hu and Evans [6] propose a scheme that provides authentication and integrity which is secure even when some nodes are compromised; it fails however in the case where two consecutive aggregators are compromised. Furthermore, this scheme addresses neither confidentiality nor
availability. Wagner [7] studies the inherent security of some aggregation functions, but he only considers the level of impact a compromised sensor may have on the final result. His work concerns the security of aggregation functions, not the aggregation security itself. Li et al. [8] already studied the use of multiple paths in the context of sensor networks security, but their work did not concern data aggregation.

III. PROBLEM FORMULATION

In the following, we describe the problems, goals, and assumptions addressed in this paper. The section is composed of two parts: (a) security aspects, and (b) network assumptions.

A. Security goals and threats

Ideally, one would like the network security to degrade gracefully with the number of compromised nodes. We consider three types of attack and assume that some link-level security mechanism is implemented to protect the network against these attacks in the absence of node compromise.

Eavesdropping. Eavesdropping occurs when an attacker compromises an aggregator node and listens to the traffic that goes through it without altering its behavior. Since an aggregator node processes various pieces of data from several nodes in the network, it does not only leak information about a specific compromised node, but from a group of nodes.

Data tampering and packet injection. A compromised node may alter packets that go through it. It may also inject false messages. Since an aggregate message embeds information from several sensor nodes, it is more interesting for an attacker to tamper with such messages than simple sensor readings. An attacker that controls the meaning of the malicious messages it sends may heavily impact the final result computed by the sink.

Denial of service. A compromised node may stop aggregating and forwarding data. Doing so, it prevents the data sink from getting information from several nodes in the network. If the node still exchanges routing messages despite its unfair behavior, that problem may be difficult to solve. Smarter attacks also involve dropping messages randomly. It is also difficult to detect when an attacker sends garbage messages.

B. Network assumptions

We assume that each sensor possesses multiple paths toward the sink and has link-level encryption capabilities. A node can then split a flow into several distinct sub-flows and send each one of them securely toward the sink. This paper does not address the problem of how to obtain multiple paths. The reader is invited to refer to the excellent literature in the domain (e.g., [9]).

We also assume that nodes have very limited computation, memory, and storage capabilities. This makes many cryptographic algorithms and protocols impractical. The schemes we propose in this paper were designed to work under such constraints.

IV. RESISTING AGAINST AGGREGATOR COMPROMISES: PROPOSED SCHEMES

We propose three schemes to achieve secure aggregation in sensor networks: Secret Multipath Aggregation (SMA), Dispersed Multipath Aggregation (DMA), and Authenticated Dispersed Multipath Aggregation (A-DMA).

All the three proposed schemes rely on the same basic principle: a sensor node splits its readings into several shares and sends these shares over distinct paths. Each share makes its way to the data sink. During forwarding, a share may be processed by aggregator nodes. Once the sink has gathered enough shares for a given set of readings, it can then reconstruct this specific set of readings. Note that a share alone is not intelligible to an intermediate node. Fig. 1 illustrates this situation.

The main property of these schemes is that the number of shares transmitted and the number of shares required for reconstructions are not necessarily equal, which means that the system inherently tolerates some losses.

A. Scheme 1: Secret Multipath Aggregation (SMA)

SMA applies secret sharing to create shares, which is a common approach when dealing with security under the contingency of node compromise.

Assume a node $i$ uses $p$ distinct paths to reach the sink, $t−1 (1 ≤ t ≤ p)$ of which may be compromised (i.e., a node must have at least $t$ shares to reconstruct the reading). Upon reading a value $r_i$, sensor node $i$ chooses a random $t−1$ degree polynomial $P_i(x)$ such that $P_i(0) = r_i$. One may construct such a polynomial by randomly choosing $a_{t,k}, \forall k \in [1, t−1]$ and using $P_i(x) = r_i + a_{t,1}x + a_{t,2}x^2 + \ldots + a_{t,t−1}x^{t−1}$. This is a simple and practical operation. Each of the $p$ shares is then composed of the values $P_i(q) (1 ≤ q ≤ p)$. Node $i$ sends then a message containing $P_i(q)$ along each path $q$.

In order to recover $r_i$, one must first recover $P_i$ using polynomial interpolation and then compute $r_i = P_i(0)$. This operation requires at least $t$ distinct shares. There is an infinity of $t−1$ degree polynomials that pass through $t−1$ points. Thus, $t−1$ compromised nodes cannot guess anything about $P_i$ and $r_i$. Note that the sink may tolerate up to $p−t$ non-responding nodes and still be able to recover $r_i$. Therefore, this scheme provides confidentiality and robustness against denial
of service attacks even in the presence of a few compromised nodes. Also, note that the sink node does not need to make any assumptions concerning the set of contributing nodes. That is, only a subset of the network’s sensor nodes may start an aggregation process.

When possible, intermediate nodes aggregate shares. Assume a node along a path must merge the readings of \( i \) and \( j \), namely \( r_i = P_i(0) \) and \( r_j = P_j(0) \). Being on path \( q \), the only data it receives is \( P_i(q) \) and \( P_j(q) \). It forwards \( P_i(q) + P_j(q) = (P_i + P_j)(q) \). The same operation is performed on the other shares of these nodes over the different paths. By receiving \( t \) samples, the sink may then recover \( P_i + P_j \) and then \((P_i + P_j)(0) = r_i + r_j \). Observe that the result also holds for multiplication by a constant.

Due to the inherent property of secret sharing, SMA offers very strong confidentiality. An attacker that has not gathered at least \( t \) shares cannot guess anything about the sensor readings. The confidentiality assured by SMA is obtained at the cost of some overhead in data transmission and therefore energy consumption. Upon sensing an event, \( p \) messages need to be sent, each one of them being of the same size as the original reading. This is the main reason for which two other schemes (DMA and A-DMA) are proposed.

### B. Scheme 2: Dispersed Multipath Aggregation (DMA)

Information dispersal is a common technique used to introduce redundancy and protection against Byzantine failures. Like secret sharing, it consists of a scheme that makes \( p \) shares out of a particular data, such that \( t \) of them are needed to reconstruct the data. Unlike secret sharing, a data block of \( t \) words is split into \( t \) pieces of one word. In our schemes, a word has the size of an aggregate.

Each sensor is pre-loaded with the same \( t \times p \) matrix \( A = [a_{i,q}] \). A should be chosen in such a way that every combination of \( t \) columns should form an invertible \( t \times t \) matrix. Note that this scheme does not rely on \( A \)'s secrecy, so sensors may embed it despite risking compromises. When sensing events, a sensor \( i \) accumulates its readings into an internal buffer of length \( t \), \( R_i = \{r_{i,1}, r_{i,2}, \ldots, r_{i,t}\} \). This forms a block of readings. Once the buffer is full, the node is ready to compute \( p \) different shares of unitary length to send along the paths. These shares are the different elements of \( M = R_i \times A \).

When the sink receives \( t \) shares, it is in position of reconstructing the data. Assuming it receives \( M_i = \{m_{i,q_1}, m_{i,q_2}, \ldots, m_{i,q_t}\} \), readings are obtained by resolving:

\[
\begin{align*}
    &\sum_{k=1}^{t} r_{i,k}a_{1,q_1} + r_{i,k}a_{2,q_1} + \ldots + r_{i,k}a_{t,q_1} = m_{i,q_1} \\
    &\sum_{k=1}^{t} r_{i,k}a_{1,q_2} + r_{i,k}a_{2,q_2} + \ldots + r_{i,k}a_{t,q_2} = m_{i,q_2} \\
    &\vdots \\
    &\sum_{k=1}^{t} r_{i,k}a_{1,q_t} + r_{i,k}a_{2,q_t} + \ldots + r_{i,k}a_{t,q_t} = m_{i,q_t}
\end{align*}
\]

(1)

This may be done using a simple Gauss elimination method or by inverting the matrix constituted of the different \( q_1, \ldots, q_t \) columns of \( A \). If the matrix \( A \) is randomly chosen, no known methods exist to reconstruct parts of the original data from \( t-1 \) samples, although some correlation between the various \( r_{i,q} \) may be deduced.

As secret sharing, this scheme permits data aggregation. Given messages \( m_{i,q} \) and \( m_{j,q} \) sent by nodes \( i \) and \( j \) on path \( q \), an aggregator node computes \( m_{i,q} + m_{j,q} \).

Since one has:

\[
    m_{i,q} + m_{j,q} = \sum_{k=1}^{t} (r_{i,k} + r_{j,k})a_{k,q},
\]

(2)

then, upon reception of at least \( t \) such messages, the sink can reconstruct every \( r_{i,k} + r_{j,k} \) in a way similar to the system shown in Eq. 1:

\[
\begin{align*}
    (r_{i,1} + r_{j,1})a_{1,q_1} + \ldots + (r_{i,t} + r_{j,t})a_{t,q_1} &= m_{i,q_1} + m_{j,q_1} \\
    (r_{i,1} + r_{j,1})a_{1,q_2} + \ldots + (r_{i,t} + r_{j,t})a_{t,q_2} &= m_{i,q_2} + m_{j,q_2} \\
    \vdots \\
    (r_{i,1} + r_{j,1})a_{1,q_t} + \ldots + (r_{i,t} + r_{j,t})a_{t,q_t} &= m_{i,q_t} + m_{j,q_t}
\end{align*}
\]

(3)

DMA is space-efficient, i.e., reconstructing \( t \) readings requires only \( t \) shares of the same size. Using more shares \((p > t)\) allows however for protection against DoS attacks.

Although the scheme is more efficient in terms of overhead than SMA, one must keep in mind that this scheme offers a weaker confidentiality than SMA. Compromised nodes allow an attacker to get some information about readings, even though partial readings cannot be reconstructed. This provides however sufficient confidentiality for sensor networks. One may therefore use this scheme to ensure loose confidentiality and resistance to node failures or DoS attacks. Note that no heavy computations are required; only the sink has to solve the system of equations (which is straightforward with any standard modern computer). Furthermore, as with SMA, the sink does not need to know the set of contributing nodes. Partial aggregation remains possible.

### C. Scheme 3: Authenticated Dispersed Multipath Aggregation (A-DMA)

SMA and DMA do not ensure protection from replay attacks nor data authenticity. A malicious attacker may eavesdrop and then send duplicates of the listened messages later. A malicious aggregator can also send garbage bits instead of the result of an expected computation and still remain unnoticed. Of course, the sink may detect such an attack by performing two reconstructions with different sets of shares and notice the results are different. But it cannot decide which of the results is correct.

There exist techniques for verifiable secret sharing but they are currently impractical for sensor networks. For this reason, we focus on an authentication solution for DMA. We propose to replace the last reading of \( R_i \) with an element that includes sequence information and depends upon a secret shared among \( i \) and the sink. Let us assume \( R_i = \{r_{i,1}, \ldots, r_{i,t-1}, h(k_i, s)\} \), where \( h(\cdot) \) is a secure hash function, \( k_i \) is a secret key between \( i \) and the sink, and \( s \) is a sequence number.

After the reconstruction of \( \sum R_i \), the sink just needs to verify whether its last element is equal to \( \sum h(k_i, s) \). If not,
then an aggregator node is assumed to be cheating and the sink has to use another subset of messages to reconstruct \( \sum R_i \). This is not to be considered as a strict integrity check because the authentication value \( h(k_i, s) \) does not gather information from the readings \( r_{i,k} \). Therefore, an attacker that has compromised \( t \) nodes might be able to reconstruct \( h(k_i, s) \) and tamper with the data without being noticed. But using information from \( r_{i,k} \) in the authentication value is not possible because the sink does not know every \( r_{i,k} \); it only reconstructs \( \sum r_{i,k} \).

Another characteristic of this scheme is that the sink needs to know the set of contributing nodes, unlike SMA and DMA. This is not really an issue when performing aggregation for full sets of shares (i.e., all of the nodes contribute). When only some of the nodes contribute to the aggregation process, however, some extra mechanisms must be provided to ensure that the sink knows the set of contributing nodes.

\[ \text{V. SECURITY ANALYSIS} \]

Security analysis in our case must be done with respect to the number of node compromises. Three fundamental questions arise:

1) How many compromised nodes does an attacker need at best to eavesdrop successfully and break confidentiality for a given scheme? Also, which nodes should be attacked?

2) What is the minimal number of nodes an attacker needs to compromise to inject false data into the network? Which nodes should be chosen?

3) How many nodes must be compromised in order for an attacker to succeed in a DoS attack?

It is important to underline that an attacker might not have the choice of which nodes to compromise. In practice, if \( n \) nodes need to be compromised for an attack to succeed, the attacker may not have access to all of these \( n \) nodes. Also, if the attacker does not have full knowledge of the topology, it may also be difficult to guess the interesting nodes to compromise. It may be a requirement that an attacker needs to compromise more nodes than the theoretical threshold.

We assume without loss of generality that only one share is sent per path. In order to show theoretical bounds, we also assume disjoint multipath routing. Note this is not a technical restriction: our schemes would perfectly work with non-disjoint paths. Of course, the more an aggregator shares paths, the more it is interesting to an attacker. No existing methods are known, however, to reconstruct parts of the readings from a subset of less than \( t \) shares.

2) Possibility of data reconstruction with \( t \) compromised paths: For the reconstruction operation to properly work, share aggregates should contain contributions from the same nodes. However, shares propagate on different paths, and among these paths the aggregator nodes receive contributions from various potentially different nodes. This makes eavesdropping attacks difficult to implement.

Consider the example of Fig. 2. Suppose that \( t = 2 \) and that an attacker has succeeded in compromising two nodes \( i \) and \( j \) on two distinct paths. If the shares gathered by \( i \) contain contributions from, say, nodes \( k \) and \( l \) and shares gathered by \( j \) contain only contributions from node \( k \), then any reconstruction will be impossible, though the attacker has compromised two distinct paths. If shares gathered by \( j \) had contained contributions from both nodes \( k \) and \( l \) then the reconstruction would have been possible. Notice that, since the data sink is an endpoint for every path, its shares gather contributions from all the nodes. It therefore does not suffer from that problem.

\[ \text{B. Data tampering and packet injection} \]

It is clear that an attacker that has compromised less than \( t \) aggregator nodes has no effective control over the signification of the data it injects into the network. Even if the attacker manages to compromise \( t \) nodes, nothing guarantees that the sink would use the shares of all those \( t \) nodes to perform a reconstruction (it may use shares from uncompromised paths). One can also imagine a scenario where an attacker succeeds in compromising one or several nodes on each possible path. Still in this case, the attacker may not be able to control the meaning of the injected/tampered data for the reason described in V-A.2.

Finally, since A-DMA provides authentication checks, an attacker that is not capable of reconstructing some readings (and therefore the authentication value for each sequence) has little chance of being able to fool the sink with tampered data. This because (a) the attacker does not know the expected authentication values and (b) it is extremely difficult for the attacker to inject a share that would modify the readings but not the authentication value after reconstruction.
attractive performance of our schemes. We compare them with our schemes in terms of both communication overhead and resistance to attacks. We then present the implementation details and simulation results that confirm the attraction of its falsifications.

C. Denial of service attacks

There are two types of DoS attacks: those where attackers stop emitting data (let us call it no-data DoS attacks) and those where they send garbage data (let us call it garbage-data DoS attacks). Note that no-data DoS attacks include the case where there is no attacker, but a sensor node simply goes down (e.g., because it runs out of battery power.) Garbage-data DoS attacks are more difficult to handle. In the absence of data authentication, an attacker needs only to compromise one path and send some garbage data on it. In this case, the sink has multiple possible outputs for reconstruction but cannot tell which ones are valid. In the presence of data authentication, garbage-data DoS attacks are indistinguishable from no-data DoS attacks – invalid reconstructions are rejected as if the wrong share had never arrived.

No-data and garbage-data DoS attacks in the presence of authentication need to prevent the sink from gathering valid shares. Therefore, an attacker needs to compromise at least $p-t+1$ distinct paths, i.e., in the worst case, $p-t+1$ nodes. If the attacker does not know the routing topology, it cannot do anything but compromise random nodes. Therefore, it will probably have to compromise more than $p-t+1$ nodes.

Let $t_e$ and $t_d$ be, respectively, the minimum number of compromised nodes required to eavesdrop communications and the minimum number of compromised nodes required to succeed in a DoS attack. From previous sections, $t_e = t$ and $t_d = p-t+1$. Note that the higher $t_e$, the lower $t_d$. One can make a trade-off by choosing $t \approx \frac{p+1}{2}$. Any higher values would give better resistance to eavesdropping whereas any lower values will give better resistance to DoS attacks. Making a relevant choice is difficult when $p$ is small (e.g., $p = 3$).

Table I summarizes the lower bounds on the number of compromised nodes one needs to succeed under the different attacks described above.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Protection from...</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple tree</td>
<td>eavesdropping</td>
<td>no</td>
</tr>
<tr>
<td>Tampering</td>
<td>tampering</td>
<td>yes</td>
</tr>
<tr>
<td>DMA</td>
<td>DoS attacks</td>
<td>no</td>
</tr>
</tbody>
</table>

VI. RESULTS

In this section we first present some other approaches and compare them with our schemes in terms of both communication overhead and resistance to attacks. We then present the implementation details and simulation results that confirm the attractive performance of our schemes.

A. Comparison to other approaches

The common insecure approach regarding data aggregation is to have one unique tree that spans every node. Each one of the tree’s internal nodes aggregates data from its children before forwarding them to its parent. With one message per node, this is the most communication-efficient technique despite the complete lack of security. This aggregation method is referred to as ‘simple tree’ hereafter, and we use it as a reference benchmark. With no overhead, one can use special encryption techniques that provide some confidentiality and still allow for aggregation to be performed [3]. One can also add different authentication mechanisms, but at the cost of larger messages [6]. Table II summarizes the features of all these schemes. As one can see, multipath aggregation schemes provide more protection against node compromises.

**TABLE I**

LOWER BOUNDS ON THE NUMBER OF COMPROMISED NODES ONE NEEDS TO SUCCEED IN VARIOUS ATTACKS.

**TABLE II**

SCHEMES’ FEATURES.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Protection from...</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMA</td>
<td>eavesdropping</td>
<td>no</td>
</tr>
<tr>
<td>DMA</td>
<td>tampering</td>
<td>yes</td>
</tr>
<tr>
<td>A-DMA</td>
<td>DoS attacks</td>
<td>no</td>
</tr>
</tbody>
</table>

A sensor node sends $p$ messages each time it does a reading. The simple tree scheme generates only one message per reading. Therefore, the overhead of the SMA scheme is $p-1$ per reading.

1) **A-DMA:** A sensor node with the A-DMA scheme sends $p$ messages each time it does $t-1$ readings. With a simple tree scheme, it would send $t$ messages. Thus, the overhead of A is $\frac{t^2}{t^2-1}$ per reading. Note that the overhead is null when $p = t$. This corresponds to the situation where all the shares are needed to reconstruct readings (i.e., no protection against DoS attacks).

2) **SMA:** A sensor node sends $p$ messages each time it does $t$ readings. With simple tree, it would send $t$ messages. Thus, the overhead of SMA is $\frac{t^2}{t}$ per reading. Note that the overhead is null when $p = t$. This corresponds to the situation where all the shares are needed to reconstruct readings (i.e., no protection against DoS attacks).

3) **DMA:** A sensor node with the DMA scheme sends $p$ messages each time it does $t-1$ readings. With a simple tree scheme, it would send $t-1$ messages. Therefore the overhead of the DMA scheme is $\frac{t^2}{t-1}$ per reading. The minimal overhead is obtained for the minimal value of $p$, that is $p = t$. As with the DMA scheme, this corresponds
to the situation where all the shares are needed to reconstruct readings.

Choosing large values of \( t \) helps reducing the overhead. One may choose \( p = \alpha t \) for a given \( \alpha \geq 1 \). This would ensure that at least \( t(\alpha - 1) \) nodes could be compromised and still remain robust to DoS attacks. The overhead then becomes \( \frac{\alpha - 1 + 1 / t}{T} \). This means that the larger \( t \), the closer the overhead to \( \alpha - 1 \) (overhead for the DMA scheme).

C. Implementation

In the following we detail the implementation of the three proposed aggregation schemes. These implementations should be seen as proof-of-concept for the feasibility of the proposed schemes, not as complete turn-key solutions.

1) Setup: Custom implementations of SMA, DMA, and A-DMA have been developed for Crossbow MICAz motes [10] running TinyOS. The operations are performed over customizable prime integer fields \( GF(p) \) and we used multi-precision computation routines from TinyECC. The source codes of the implementations can be downloaded from http://www-rp.lip6.fr/~claveiro/secure-aggreg/.

For the sake of simplicity and in order to isolate our results from any bias introduced by the routing layer, we used optimized static multipath routing (made by hand). This layer uses the default TinyOS link layer, which is not secure enough with regard to the assumptions taken in this paper. However, thanks to TinyOS modular design, one can write and use his own layers for routing and secure-link establishment without being intrusive.

Once compiled, many parameters impact memory occupancy. Let us consider the size of a \( GF(p) \) integer or the information dispersal \( A \) matrix size. Fig. 3 presents the memory footprints for some typical parameters. SMA’s footprint is rather good whatever integer field is used (about the half of a MICAz’s RAM, for instance). DMA’s and A-DMA’s footprints are very sensitive to the size of the information dispersal \( A \) matrix. This explains why DMA’s and A-DMA’s curves are linear: \( A \)’s size is linear with integer sizes. This matrix determines the maximum number of shares \( p \) and threshold \( t \) of DMA and A-DMA schemes. Some large values, such as \( 16 \times 16 \) matrices with 64 bits integers do not fit into a MICAz mote. Other values are however fairly reasonable with respect to memory occupancy.

The time required for the nodes to perform operations such as share creation and aggregation is negligible and has never been an issue during tests. Concerning storage, each reading is only a few bytes, so buffering readings for information dispersal is not a problem either.

The aggregation processes work as follows. Nodes sense some data at regular intervals and push shares toward the sink using a sequence number. An aggregator node only aggregates shares having the same sequence number. When an aggregator node receives a share, it stores the share in a buffer and waits (during the period of a timer) for other shares with the same sequence number.

2) Experimentations: We perform both real experiments and simulations using the implementation described above. Experiments are done at small scale (six nodes and three-path topologies) to test the practicality of the schemes. In order to stress the implementation, we also perform large-scale simulations using TOSSIM.

Our goal is to evaluate the overhead in terms of energy consumption introduced by our implementations (it is important
to keep in mind that security necessarily implies a cost). To this end, we measure the number of messages sent for different implementation parameters and schemes (see Fig. 4). For each scheme and each \((p, t)\) we run the simulation five times. Each run uses a different topology of forty nodes with one-hour simulation period. This lowers the impact a specific topology may have on the number of messages transmitted. Each point represents the total number of messages sent during all runs. The goal is to illustrate the behavior of our schemes when compared with simple-tree. The topologies use sink-rooted node-disjoint trees to perform multipath routing. Two topologies have four paths, the others have respectively three, six, and eight paths. Note that the number of paths has no impact on the number of messages sent (which is only influenced by \(p\) and \(t\)). When there are more shares than available paths, some paths carry multiple shares. During simulations, we use TOSSIM’s standard random number generator.

We can observe some predictable properties of the schemes. As previously analyzed, SMA’s overhead is the highest one, depending solely on \(p\). Therefore, SMA roughly needs \(p\) additional messages compared to the simple tree scheme. We can also see that the overhead of information dispersal based schemes depends on \(\alpha = \frac{t}{p}\). It is not a surprise that A-DMA and DMA exhibit similar performance, with DMA having a slightly lower overhead. Overheads are however a bit higher than computed in section VI-B. As an example, A-DMA with \(p = 12\) and \(t = 8\) has an overhead of 1 instead of the predicted 0.7. Also, A-DMA with \(p = t = 12\) exhibit a small overhead of 0.25 instead of about 0.1. This is due to the practical considerations (e.g., timer values) that make implementations miss some aggregation opportunities.

VII. CONCLUSION AND FUTURE WORKS

We proposed three schemes to secure data aggregation using multipath routing. They are based on secret sharing and information dispersal. In the proposed schemes, sensors split their readings into several shares and distribute them among several disjoint paths. Upon reception of a minimum number of shares, the sink can reconstruct the aggregated value.

Depending on the scheme and its parameters, these techniques provide varying levels of resistance against DoS attacks, eavesdropping, and data tampering. By using secret multipath aggregation, one can guarantee that a subset of compromised paths cannot reveal/leak any information about the readings. This is at the cost of some overhead. By using dispersed multipath aggregation, one has an optimal overhead but achieves lower levels of confidentiality. Depending on the application or scenario, one approach offers more advantages than another.

To the best of our knowledge, the three proposed schemes are the first to address node compromises for aggregation schemes in sensor networks using multiple paths. Future work includes modeling the security parameters’ statistical behavior under the contingency of random node compromises. We would also like to generalize and apply these schemes to contexts other than sensor networks, e.g., delay tolerant networking or ad hoc networking.

REFERENCES