Institut Eurécom
2229, route des Cretes
B.P. 193
06904 Sophia Antipolis
FRANCE

Research Report RR-06-181

Secure data aggregation with multiple encryption

March 2006

Melek Önen, Refik Molva

Tel : (+33) 4 93 00 81 00
Fax : (+33) 4 93 00 82 00
Email : {onen, molva}@eurecom.fr

[2] This research is supported by Institut Eurcom's industrial members: BMW, Bouygues Tlcom, Cisco Systems, France Tlcom , Hitachi Europe, SFR, Sharp, STMicroelectronics, Swisscom, Thales.

Abstract

Data aggregation has been put forward as an essential technique to achieve power efficiency in sensor networks. Data aggregation consists of processing data collected by source nodes at each intermediate node enroute to the sink in order to reduce redundancy and minimize bandwidth usage.

The deployment of sensor networks in hostile environments call for security measures such as data encryption and authentication to prevent data tampering by intruders or disclosure by compromised nodes. Aggregation of encrypted and/or integrity-protected data by intermediate nodes that are not necessarily trusted due to potential node compromise is a challenging problem. We propose a secure data aggregation scheme that ensures that sensors participating to the aggregation mechanism do not have access to the content of the data while adding their sensed values thanks to the use of an efficient homomorphic encryption scheme. We provide a layered secure aggregation mechanism and the related key attribution algorithm that limits the impact of security threats such as node compromises. We also evaluate the robustness of the scheme against node failures and show that such failures are efficiently recovered by a small subset of nodes that are at most m hops away from the failure.

1 Introduction

Wireless sensor networks (WSN) are viewed as a popular solution to various monitoring problems such as safety monitoring, wildfire tracking and traffic monitoring. A WSN consists of thousands of sensors that are in charge of both monitoring and data transmission tasks. The data collected by each sensor is transmitted via a network consisting of other sensors towards a well identified destination node called sink. In the basic setting of a WSN, each individual piece of data is thus independently transmitted over several hops towards the sink and each sensor node is involved in the forwarding of a large number of data pieces originated from other sensors. In the resource constrained WSN environment, forwarding of large amounts of data becomes the major focus of energy and bandwidth optimization efforts. Data aggregation has thus been put forward as an essential technique to achieve power and bandwidth efficiency in WSN. Based on the principle that the sink does not necessarily need all raw pieces of information collected by each sensor but only a summary or aggregate thereof, data aggregation consists of processing data collected by source nodes at each intermediate node enroute to the sink in order to reduce redundancy and minimize bandwidth usage. A common way to aggregate data in sensor networks is to simply sum up values as they are forwarded towards the sink. Such additive aggregations are useful for statistical measurements such as mean or variance computation.

As a distributed task achieved by several potentially compromised nodes, data aggregation raises some new security concerns in addition to the basic vulnerabilities of a WSN [**?**]. Data aggregation in WSN is thus exposed to various threats such as node compromise, injection of bogus aggregates, disclosure of sensed data and aggregate values to intruders or tampering with data transmitted over wireless links. In this paper, we focus on the problem of data confidentiality with a twofold objective: first to prevent intruders from accessing individual monitoring results, second to prevent any node other than the sink from accessing the aggregate values. While classical data encryption mechanisms easily meet the first objective, the second objective raises a new requirement for sensor nodes involved in the computation of intermediate aggregate values: each sensor node must be able to combine the locally monitored value that is in cleartext with the encrypted aggregate value received from adjacent nodes in order to come up with a new encrypted aggregate value. This problem typically calls for some form of homomorphic encryption technique. Existing solutions based on homomorphic encryption [**?**, **?**] either suffer from excessive computational complexity or are vulnerable to node compromise.

We suggest a secure additive data aggregation scheme based on the use of an efficient homomorphic encryption technique combined with a multiple encryption scheme using symmetric algorithms. The homomorphism of the underlying encryption technique allows sensors to aggregate their cleartext measurements with the encrypted aggregate values whereas the multiple encryption scheme assures that aggregate values and individual measurement results remain oblivious to all intermediate nodes enroute to the sink. The joint use of the homomorphism and multiple encryption assures that a secret channel is established between every sensor node and the sink without having to establish pairwise security associations or a public-key infrastructure.

We first analyze the security requirements raised by secure data aggregation and describe the need for homomorphic encryption functions. We then briefly present the CTRM encryption scheme proposed by Bellare et al. in [**?**], and its extension in [**?**] for the context of multicast confidentiality. We show that CTRM is homomorphic and introduce the proposed layered secure aggregation scheme based on CTRM. We then evaluate the effectiveness of the proposed scheme in terms of security, safety and performance.

2 Problem Statement

2.1 Aggregation in Wireless Sensor Networks

We model a wireless sensor network (WSN) as a rooted tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of nodes corresponding to the sensors and $\mathcal{E}$ is the set of edges between these nodes. The root S of the tree corresponds to the sink. Each other node has one or more incoming edges but a unique outgoing edge.

Aggregation techniques are used to reduce the amount of data communicated within a WSN. As measurements are recorded periodically at each sensor, one way to aggregate such information is the additive aggregation that is the addition of values as they are forwarded towards the sink. Each node receives packets from the incoming edges, aggregates them and sends the result via the outgoing edge. The sink collects the final set of aggregated packets and completes the aggregation task. Additive aggregation techniques are very useful for statistical measurements in sensor networks. Hence, once the sink receives the addition of some values, it can easily compute the mean or variance of the received values.

2.2 Security requirements

In the context of secure data aggregation, we distinguish two confidentiality requirements:

- **generic confidentiality** whereby sensors not participating to the aggregation mechanism, should not have access to the content of the data.
- **end-to-end confidentiality** whereby sensors actively participating to the aggregation mechanism do not access the data that is already aggregated.

As to generic confidentiality, sensors need to use some cryptographic encryption algorithms in order to let only authorized sensors access the content of the data. Since sensor nodes have very limited resources, symmetric encryption algorithms are more suitable for such networks. However, with the use of classical encryption schemes such as AES [**?**], every sensor should first decrypt the received measurements in order to aggregate their own measured value and then re-encrypt the result in order to send it to the next sensor enroute to the sink. In this case, all sensors would have access to aggregated measurements. In order to prevent such access and thus to ensure end-to-end confidentiality, we propose a new framework that implements homomorphic encryption algorithms.

2.3 The proposed framework

We propose a framework whereby sensors participate to a secure aggregation mechanism without having access to the protected data. In order to ensure end-to-end confidentiality, the framework uses additive homomorphic encryption algorithms. Moreover, measurements are protected with multiple encryption layers. Sensors receiving encrypted data would be able to suppress some encryption layers, aggregate their measurements and add new encryption layers. Thanks to a new key attribution algorithm, only the sink is able to suppress all encryption layers and thus access the finally aggregated result. Since each sensor modifies the encryption of the data, the compromise of some intermediary nodes does not provide access to the protected data.

In the following section, we describe the CTRM homomorphic encryption algorithm that is extended in our framework. We then introduce the new key attribution algorithm that is used in the new secure aggregation scheme that ensures both generic and end-to-end confidentiality.

3 The proposed encryption algorithm

3.1 Additive homomorphic encryption

End-to-end confidentiality as defined in section 2.2 requires a homomorphic encryption scheme. A homomorphism is defined as a map $\phi : X \longrightarrow Y$ such that:

$$\phi(x \cdot y) = \phi(x) \circ \phi(y) \tag{1}$$

where $\cdot$ and $\circ$ respectively are the operations in X and Y. If ϕ is a homomorphic encryption algorithm, and if $\cdot$ is the aggregation operation, thanks to the homomorphism of ϕ encrypted individual measurements can be aggregated into an encrypted aggregate value. Hence, let N_i be a sensor receiving encrypted measurements $\phi(V_j)$ and $\phi(V_k)$. N_i first senses V_i, computes $\phi(V_i)$ and aggregates the three encryptions as $\phi(V_j) \circ \phi(V_k) \circ \phi(V_i)$. Thanks to the homomorphism of ϕ, this result is identical to the encrypted aggregate value: $\phi(V_j \cdot V_k \cdot V_i)$. It should be noted that N_i was able to aggregate its measurement with the received values without accessing the measurements in this example.

Let $\mathcal{M}$ be the set of plaintext messages and $\mathcal{K}$ be the set of encryption keys. In the context of secure data aggregation, we propose that the encryption function

ϕ represents an additive homomorphic encryption scheme that encrypts a message $x \in \mathcal{M}$ with the encryption key $k \in \mathcal{K}$ as follows:

$$\begin{aligned} \phi &: (\mathcal{M}, \mathcal{K}) \longrightarrow \mathcal{M} \\ \phi(x, k) &= (x + k) \bmod n \end{aligned} \tag{2}$$

n is the cardinality of $\mathcal{M}$. It is easy to show that ϕ is a homomorphic function. Hence, let x_a and x_b two different plaintext messages in $\mathcal{M}$ and k_a and k_b encryption keys in $\mathcal{K}$. We have:

$$\begin{aligned} \phi(x_a, k_a) &= (x_a + k_a) \bmod n \\ \phi(x_b, k_b) &= (x_b + k_b) \bmod n \\ \phi(x_a, k_a) + \phi(x_b, k_b) &= (x_a + k_a + x_b + k_b) \bmod n \\ &= \phi(x_a + x_b, k_a + k_b) \end{aligned}$$

The security of this scheme relies on the unique utilization of the key. Hence, as one-time pads, for each message, the encryption must use a different key. Thus, an efficient key generation algorithm is required for each encryption operation. We propose to implement the basic CTRM function proposed by Bellare et al. in [**?**] that allows the generation of a different key for each encryption operation. Thanks to this scheme that is briefly described in the following section, sensors are able to update their encryption key without receiving any additional information from the sink.

3.2 CTRM encryption scheme

In [**?**], Bellare et al. describe and analyze various cipher modes of operation. In this section, we briefly describe their proposed counter based block cipher mode of operation (CTR-mode) which we extend in our proposed scheme. We denote $\oplus$ as the binary XOR operation and define f_a as a l-bit pseudorandom permutation such as AES [**?**] where a is the encryption key. The CTR-mode scheme is a triplet $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ defined as follows:

- $\mathcal{K}$ flips coins and outputs a random key a;
- $\mathcal{E}(ctr, x)$ splits x into n blocks of l bits $x = x_1, .., x_n$, and for each x_i returns $y_i = f_a(ctr + i) \oplus x_i$. Finally, ctr is updated by $ctr + n$;
- Symmetrically, $\mathcal{D}(ctr, y)$ first splits y into n blocks of l bits $y = y_1, .., y_n$, and for each y_i, it returns $x_i = f_a(ctr + i) \oplus y_i$. Similarly, ctr is updated by $ctr + n$.

The counter ctr is maintained by the encryption algorithm across consecutive encryptions with the same key. Thanks to this counter, the receiver that knows the key a can recompute each $f_a(ctr + i)$ and thus retrieve the original message x.

3.3 Multiple key CTRM encryption for secure data aggregation

In order to introduce the secure data aggregation, we propose an extended version of the CTRM encryption with the use of multiple keys for both encryption and decryption. We first replace the XOR operation by the additive homomorphic encryption scheme defined in equation 2. In the sequel of this paper $a + b$ and $a - b$ are respectively defined as $(a + b) \bmod n$ and $(a - b) \bmod n$. The new basic encryption is again a triplet $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ such that:

- $\mathcal{K}$ flips coins and outputs a random key a;
- $\mathcal{E}(ctr, x)$ splits x into n blocks of l bits and for each x_i returns $y_i = f_a(ctr + i) + x_i$. Finally ctr is updated by $ctr + n$;
- $\mathcal{D}(ctr, y)$ splits y into n blocks of l bits and for each y_i returns $x_i = y_i - f_a(ctr, +i)$.Finally, ctr is updated by $ctr + n$.

Since $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ is also homomorphic, we now focus on the problem of end-to-end confidentiality whereby sensors perform aggregation operations using this scheme. Since sensors are not authorized to access the content of their received aggregated information, different keys should be distributed to each sensor. In this case, we propose a triplet $(\mathcal{K}^{(r)}, \mathcal{E}^{(r)}, \mathcal{D}^{(r)})$ with r independent keys as follows:

- $\mathcal{K}^{(r)}$ chooses r random keys $a_1, .., a_r$;
- $\mathcal{E}^{(r)}(ctr_1, .., ctr_r, x)$ splits x into n blocks of l bits $x = x_1, .., x_n$, and for each x_i returns $y_i = x_i + \sum_{j=1}^{r} f_{a_j}(ctr_j + i)$;
- $\mathcal{D}^{(r)}(ctr_1, .., ctr_r, y)$ splits y into n blocks of l bits $y = y_1, .., y_n$ in order to retrieve $x_i = y_i - \sum_{j=1}^{r} f_{a_j}(ctr_j + i)$.

We recall the security property that claims that a message encrypted with multiple keys is at least secure as any individual encryptions [**?**]. It is obvious that $(\mathcal{K}^{(r)}, \mathcal{E}^{(r)}, \mathcal{D}^{(r)})$ is homomorphic since the encryption and decryption operations are respectively defined by additions and subtractions that are by definition homomorphic.

4 The proposed model: layered secure aggregations

Now that we have defined the security requirements specific to the problem of data aggregation and that we have described the proposed CTRM encryption algorithm, we describe the proposed layered secure aggregation scheme that allows sensors to aggregate measurements while the data remains confidential. Thanks to the addition of multiple encryption layers, the scheme remains secure against attacks such as node compromise. We first introduce a new key attribution algorithm that defines the keying material of each sensor and then present the aggregation protocol.

4.1 Notation

As described in section 2.1, a wireless sensor network is represented by a tree $\mathcal{T}$. We define the function $Depth$ that given a node identity N_i returns its depth in the tree. We set $Depth(S) = 0$. Within this tree, we also define the following relations between nodes:

- $Root(\mathcal{T})$ represents the data sink that collects and extracts the aggregated data;
- $Parent(N, m)$ is the mth parent of N if it exists or S otherwise;
- $Children(N, m)$ is the set of nodes N_i such that $\forall i, N = Parent(N_i, m)$.

In order to implement the CTRM encryption algorithm with multiple encryption keys in the context of secure data aggregation, we define a key attribution algorithm that is explained in the following section. Thanks to this algorithm, any node will be able to add or suppress some encryption layers without causing any leakage of secret information.

4.2 The proposed key attribution algorithm

In this section, we describe a new key attribution algorithm for the proposed aggregation protocol. Thanks to this algorithm, the sink is able to aggregate all the measurements without leaking any secret information to any node including the sensors that participate to the aggregation mechanism.

Each node N_i shares a key $a_{i,j}$ and a counter $ctr_{i,j}$ with a node N_j where $N_j = Parent(N_i, m)$. We also define a different key and counter $(a_{l,k}, ctr_{l,k})$ shared between a leaf node N_l and $N_k = Parent(N_l, t)$, for each $0 < t < m$. The key attribution algorithm is summarized in Table 1.

In order to illustrate this algorithm, we define a WSN with 11 nodes represented in Figure 1. In this particular network, we set $m = 2$. Following the key attribution protocol, all leaf nodes, N_5, N_6, N_9 and N_{10} share one key with their direct parent and another one with their grandparent. For example, node N_9 shares $a_{7,9}$ with node N_7 and $a_{4,9}$ with node N_4. Node N_1 which is an intermediate node, shares a different key with nodes N_5, N_6, N_7 and N_8 which are in $Children(N_1, 2)$ and with S since $Parent(N_1, 2) = S$.

```
For each node $N_i$ in $\mathcal{T}$:
   define $(a_{i,j}, ctr_{i,j})$ for $N_i$ and $N_j = Parent(N_i, m)$;
   if $N_i$ is a leaf node
   then
       for each $t < m$
           define $(a_{i,k}, ctr_{i,k})$ for $N_i$ and $N_k = Parent(N_i, t)$;
   else
       set $t = 0$;
       while $Children(N_i, m - t) = \emptyset$
           increment t by one;
           define $(a_{i,j}, ctr_{i,j})$ for $N_i$ and $N_j \in Children(N_i, m - t)$;
```

Table 1: The key attribution algorithm

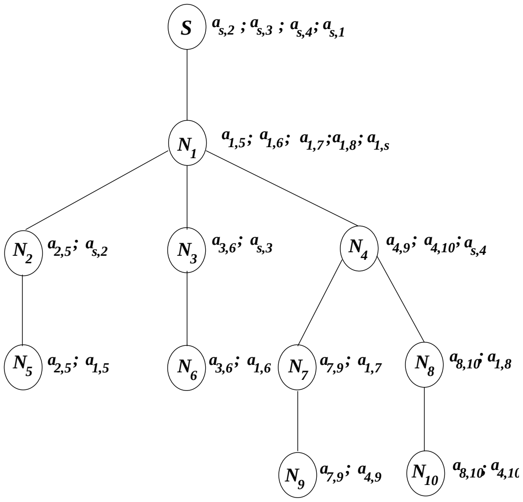

Figure 1: Implementation of the key attribution algorithm with $m = 2$

```
For each $N_i$ with measured value $V_i$

  Receive $\{A_j\}$ from $N_j \in Children(N_i, 1)$;
  Compute $S_i = \sum_{j, N_j = Parent(N_i, m)} A_j$;

   for all $l$ where $N_i = Parent(N_l, m)$
      Compute $Sd_i = D(ctr_{i,l}, S_i)$;

  Compute $A_i = \mathcal{E}(ctr_{k,i}, Sd_i + V_i)$ such that $N_k = Parent(N_i, m)$;
```

Table 2: The additive aggregation protocol

4.3 The aggregation protocol

Now that we have defined the key attribution algorithm, each node is ready to aggregate its measurement with the received values from its children nodes. In this paper, we define the aggregation operation as a sum computation. This operation can also be a mean or variance computation. Since the encryption algorithm is homomorphic, each node adds the received values to the measured value without having to access the content of the aggregated data.

Table 2 illustrates the additive aggregation protocol. A sensor N_i first aggregates the received values and its measurement. From this value, N_i subtracts keys that it shares with its mth children nodes N_j and adds the key that it shares with its mth parent node N_k. Then, N_i sends the aggregated value denoted by A_i to its parent node.

As an example, we examine in Table 3 how the proposed additive aggregation protocol is applied on the tree of Figure 1. For the sake of clarity, we define $k_{i,j}$ as the one-time-key originating from $a_{i,j}$ and $ctr_{i,j}$.

5 Evaluation

In the following sections, we review the proposed framework with respect to:

- **confidentiality** whereby intruders and sensors should not have access to the content of the data (generic and end-to-end confidentiality);

- **robustness** whereby the impact of a node compromise or a node failure on the aggregation scheme should be minimized.

We then evaluate the performance of the scheme in terms of memory and CPU usage and in terms of communication overhead.

Layer 4:

Node N_9: Computes $A_9 = V_9 + k_{7,9} + k_{4,9}$

Node N_{10}: Computes $A_{10} = V_{10} + k_{8,10} + k_{4,10}$

Layer 3:

Node N_5: Computes $A_5 = V_5 + k_{2,5} + k_{1,5}$

Node N_6: Computes $A_6 = V_6 + k_{3,6} + k_{1,6}$

Node N_7: Receives $A_9 = V_9 + k_{7,9} + k_{4,9}$

Suppresses a layer $Sd_7 = A_9 - k_{7,9}$

Computes $V_7 + k_{1,7}$

Adds a layer $A_7 = V_9 + V_7 + k_{4,9} + k_{1,7}$

Node N_8: Receives $A_{10} = V_{10} + k_{8,10} + k_{4,10}$

Suppresses a layer $Sd_8 = A_{10} - k_{8,10}$

Computes $V_8 + k_{1,8}$

Adds a layer $A_8 = V_{10} + V_8 + k_{4,10} + k_{1,8}$

Layer 2:

Node N_2: Receives $A_5 = V_5 + k_{2,5} + k_{1,5}$

Suppresses a layer $Sd_2 = A_5 - k_{2,5}$

Computes $V_2 + k_{s,2}$

Adds a layer $A_2 = V_5 + V_2 + k_{1,5} + k_{s,2}$

Node N_3: Receives $A_6 = V_6 + k_{3,6} + k_{1,6}$

Suppresses a layer $Sd_3 = A_6 - k_{3,6}$

Computes $V_3 + k_{s,3}$

Adds a layer $A_3 = V_6 + V_3 + k_{1,6} + k_{s,3}$

Node N_4: Receives A_7 and A_8

Aggregates $S_4 = A_7 + A_8$

Suppresses two layers $Sd_4 = A_7 + A_8 - k_{4,9} - k_{4,10}$

Computes $V_4 + k_{s,4}$

Adds a layer $A_4 = V_{10} + V_9 + V_8 + V_7 + V_4 + k_{1,7} + k_{1,8} + k_{s,4}$

Layer 1:

Node N_1: Receives A_2, A_3 and A_4

Aggregates $S_1 = A_2 + A_3 + A_4$

Suppresses four layers $Sd_1 = A_2 + A_3 + A_4 - k_{1,5} - k_{1,6} - k_{1,7} - k_{1,8}$

Computes $V_1 + k_{s,1}$

Adds a layer $A_1 = \sum_{i=1}^{10} V_i + k_{s,2} + k_{s,3} + k_{s,4} + k_{s,1}$

Layer 0:

Sink S: Receives A_1

Suppresses all layers $Sd_s = A_1 - k_{s,2} - k_{s,3} - k_{s,4} - k_{s,1}$

Table 3: The Additive Aggregation Protocol: an example

5.1 Security evaluation

In this section, we first show that the proposed framework ensures generic confidentiality and then consider the node compromise scenario that could prevent the end-to-end confidentiality of the scheme.

Proposition 1 *The scheme ensures generic confidentiality.*

Proof In a work evaluating the security of cryptosystems in the multi-user setting [**?**], Bellare et al. have essentially shown that if a cryptosystem is secure in the sense of indistinguishability, then the cryptosystem in the multi-user setting, where related messages are encrypted using different keys, is also secure. This result can be applied to the proposed scheme using CTRM. When a message is encrypted with r keys it is at least as secure as any individual encryption. Thus, the scheme is at least as secure as a one layer encryption layer, if no node is compromised.

Moreover the security of encryption operation that simply is a modulo n addition depends on the unique utilization of the encryption key. Thanks to the existence of a counter, at each encryption operation, the encryption key is updated and thus the operation is perfectly secure. ■

We now consider the node compromise scenario.

Proposition 2 *An intruder can have access to an aggregated data originating from node N_i only in two cases:*

- *Case 1: all the nodes in the subtree T^* of $\mathcal{T}$ whose root is N_i and depth is $m-1$ are compromised;*
- *Case 2: all nodes N_l such that $N_l = Parent(N_i, k)$ for all $1 \leq k \leq m-1$ are compromised;*

Proof Let's assume that node N_i is compromised. Then the intruder has access to all keys stored by N_i, that are:

- $\{a_{i,j}\}$ shared between N_i and N_j such that $N_i = Parent(N_j, m)$;
- $a_{i,k}$ shared between N_i and N_k such that $N_k = Parent(N_i, m)$;

When N_i receives aggregated values from its children nodes, these values are still encrypted with different keys by the nodes $N_j \in Children(N_i, k)$ with $1 \leq k \leq m$. Consequently, in addition to N_i, the intruder needs to compromise all the nodes in the subtree T^* of $\mathcal{T}$ whose root is N_i and depth is $m-1$. This proves the Case 1 of proposition 2.

Furthermore, the keys used for the encryption of aggregated values by nodes N_j that construct T^* are by definition shared with nodes N_l such that $N_l = Parent(N_j, m)$. Consequently if the intruder compromises these nodes, it also can access the aggregated data originating from N_i. Since $N_i = Parent(N_j, k)$ with $1 \leq k \leq m$, the intruder needs to compromise nodes N_l such that $N_l = Parent(N_i, n)$ with $1 \leq n \leq m-1$. This result proves the Case 2 of proposition 2.

$\blacksquare$

Therefore, the security of the scheme in terms of end-to-end confidentiality depends on the choice of the value m. The larger values for m imply a larger population to compromise for the intruders. However, if m is very large, the scheme becomes inefficient since the number of encryption layers decreases and the scheme tends to be vulnerable to threats such as node compromise. Hence, if m equals the depth of $\mathcal{T}$ denoted by h, all nodes would share one key with the sink. In this case, the advantage of the use of multiple encryption layers disappears and the proposed scheme would be similar to the secure data aggregation scheme in [**?**]. The scheme would still ensure end-to-end confidentiality, but a node failure would have a strong impact on the aggregation scheme since in addition to the aggregated data, sensors must include additional information about the identities of nodes participating to the aggregation. Thus m must not exceed $h-1$. As a result, m should be as large as possible for security reasons and small enough for the sake of robustness. The ideal value for m would be the minimum depth of all leaf nodes in $\mathcal{T}$.

5.2 Robustness of the scheme

Data aggregation in WSN is exposed to the following threats:

- **node compromise** whereby intruders can have access to the security material of a sensor participating to secure data aggregation. In this case, the aggregation scheme is exposed either to the injection of bogus aggregates or to some passive behavior from the compromised node;
- **node failure** whereby the node is off and thus cannot participate to the aggregation mechanism;
- **communication failure** whereby messages enroute to the sink are lost;
- **poisoning** whereby intruders inject some bogus data and thus break the aggregation mechanism.

The impact of a node failure or a communication failure remains the same as the impact of passive behavior originating from a compromised node. Hence, in all cases, a sensor does not receive any message from some of its children nodes and thus should exclude some of its keying material from the next aggregation process. The impact of such failures should be minimized.

Poisoning attacks and the injection of bogus aggregates by compromised nodes first imply a strong need for an authentication mechanism that allows a sensor to verify the origin and the integrity of the received data. We assume that there

```
let l = 0;
at layer l, for each $N_i \in \mathcal{T}$
  $verify(agg\_value, expected\_value)$
  if OK then
      ACCEPT $agg\_value$;
  else
      if $Children(N_i, 2) = \emptyset$ then
          $send\_alert(identity(Children(N_i)))$
      else
          send $(expected\_value, keying\_material)$ to $Children(N_i, 1)$;
```

Table 4: The discovery of compromised nodes

is an underlying authentication mechanism such as digital signatures. However, compromised nodes still can inject bogus aggregates although the verification of their signature succeeds. In this particular case, since sensors do not have access to the content of aggregates, such attacks are not detected and thus bogus messages cannot be immediately discarded. We thus propose, a recovery mechanism rather than a prevention mechanism that allows the sink to react against such attacks by determining the origin of the attacks.

We thus mainly distinguish two classes of robustness problems and come up with some recovery mechanisms for each of them: **the bogus message injection** originating from compromised nodes and **the loss of messages**.

5.2.1 Protection against bogus message injection

We first evaluate the performance of the scheme when the intruder compromising N_i performs some bogus injection. In this case, the sink might possibly notice the attack once the aggregation protocol is complete, that is, when it decrypts the aggregated value. Therefore, the sink cannot prevent such attacks but can react against them by determining the origin of the attacks. Hence, when the sink notices such attacks due to some exaggerated values that would result from aggregation, it first contacts its children nodes and sends them the required decryption material (that is one-time) in order to let them discover the origin of the failure. This process is recursively run along the tree. Thus, the cost of discovering the compromised node is in the order of $log(N)$ where N is the number of sensors and the verification task is distributed to all nodes of the tree. The process of compromised nodes discovery is summarized in Table 4.

5.2.2 Protection against message losses

We now consider the case when there is a node or communication failure that imply some message loss: an error may occur during the decryption of the aggregated data. The same problem can happen when an intruder compromising a node shows a passive behavior. In this case, a node that did not receive any aggregated information from one of its children nodes, alerts nodes that are at most m distant from it about the identity of the misbehaving node. All nodes receiving this alert, will remove the keys that are related with the misbehaving node and proceed the aggregation protocol with the remaining keys. The alert messages only reach nodes that are m distant from the misbehaving node and thus have a local impact on the communication overhead.

In order to illustrate this recovery mechanism, we again refer to the WSN represented in Figure 1 and we assume that node N_{10} did not send its measurement to N_8. For the sake of simplicity, we again denote the one-time key resulting from $a_{i,j}$ and the actual $ctr_{i,j}$ by $k_{i,j}$. In this particular case, keys $k_{8,10}$ and $k_{4,10}$ should not be used during the aggregation protocol. Thus, N_8 sends an alert message with the identity of N_{10} to N_4. Since $m = 2$ and $N_4 = Parent(N_{10}, m)$, N_4 does not need to forward this alert message to its parents. Table 5 illustrates the aggregation process for nodes that are on the path from N_{10} to the sink. While computing A_8, N_8 only includes V_8 that is encrypted with $k_{1,8}$. When N_4 receives A_8, and A_7, it does not use $k_{4,10}$ and suppresses the only encryption layer originating from node N_9 and finally adds an encryption layer with $k_{s,4}$. Once N_4 sends A_4 to N_1, there is no more modification in the aggregation process and N_1 will follow the additive aggregation protocol as defined in Table 2.

5.3 Performance evaluation

In this section, we evaluate the performance of the scheme in terms of memory storage, computational cost and communication overhead. The computational cost and communication overhead have a direct impact on the battery usage.

First of all, the computational activity of each sensor for the encryption and decryption operations is only the sum and substraction operations modulo n. The encryption or decryption operations do not have an impact on the communication overhead. There is no additional information with respect to these two operations. The sink only receives messages from its children nodes and proceeds to the final step of aggregation.

Furthermore, thanks to the inherent key generation process provided by CTRM, there is no additional overhead originating from the update of any sensor's keys.

The memory cost is related to the proposed key attribution algorithm. Sensors share one key with their mth parent node and one key with each of their mth child nodes. Furthermore, if a sensor is a leaf node of $\mathcal{T}$, this sensor shares one key with each of its kth parent with $1 \leq k \leq m$. Thus, the memory cost for each sensor equals to:

- $(|Children(N, m)| + 1)$ if $Children(N, m) \neq \emptyset$,
- $(|Children(N, k)| + 1)$ if $Children(N, k) \neq \emptyset$ and $Children(N, k+1) = \emptyset$.

```
Layer 3

  Node $N_8$ Does not receive $A_{10}$
    mark $k_{8,10}$ as invalid
    Computes $A_8 = V_8 + k_{1,8}$
    Sends $A_8$ and $failure\_alert(N_{10})$

Layer 2

  Node $N_4$ Receives, $A_7$, $A_8$ and $failure\_alert(N_{10})$
    mark $k_{4,10}$ as invalid
    Aggregates $S_4 = A_7 + A_8$
    Suppresses one layer $Sd_4 = A_7 + A_8 - k_{4,9}$
    Computes $V_4 + k_{s,4}$
    Adds a layer $A_4 = V_9 + V_8 + V_7 + V_4 + k_{1,7} + k_{1,8} + k_{s,4}$

Layer 1

  Node $N_1$ Receives $A_2$, $A_3$ and $A_4$
    Aggregates $S_1 = A_2 + A_3 + A_4$
    Suppresses four layers $Sd_1 = A_2 + A_3 + A_4 - k_{1,5} - k_{1,6} - k_{1,7} - k_{1,8}$
    Computes $V_1 + k_{s,1}$
    Adds a layer $A_1 = \sum_{i=1}^{9} V_i + k_{s,2} + k_{s,3} + k_{s,4} + k_{s,1}$

Layer 0

  Sink $S$: Receives $A_1$
    Suppresses all layers $Sd_s = A_1 - k_{s,2} - k_{s,3} - k_{s,4} - k_{s,1}$
```

Table 5: Failure recovery of N_{10} in the path from N_{10} to S

6 Related Work

In [**?**, **?**], authors propose to use homomorphic encryption schemes to allow secure data aggregation. They implement the Domingo-Ferrer encryption scheme [**?**] that is based on the computationally expensive discrete exponential technique. The feasibility of this scheme in the context of resource constrained sensor environment is analyzed in [**?**] and authors gave performance results on the Mica2 motes [**?**] and show that such measurements were quite reasonable.

In [**?**], authors propose a secure data aggregation scheme similar to ours that is based on an extension of the one-time pad encryption technique using additive operations modulo n. Even though our scheme seems to be more complex than the solution of [**?**] due to the use of CTRM and multiple encryption layers, our scheme clearly imposes a lower communication overhead than the latter. In [**?**], each aggregate message is coupled with the list of nodes that failed to contribute to the aggregation because of node or communication failures. As opposed to [**?**], in our scheme each failure only needs to be reported during m hops from the location of the failure enroute to the sink. Thus, our scheme does not require the reporting of failures beyond the mth parent of the failure point in the tree. Moreover, [**?**] overlooks the key update that is a mandatory security requirement for additive encryption schemes whereas our solution includes a complete mechanism for key update based on CTRM. Hence, the security of the additive encryption operation is based on the unique utilization of the encryption key. Since in [**?**] sensors only share keys with the sink, at each aggregation operation, all keys that are used for such operation should be marked as invalid and separately updated by the sink. In our proposed scheme, thanks to the use of CTRM, sensors are able to update their keys without receiving any additional message from the sink.

7 Conclusion

In this paper, we analyze the problem of confidentiality in secure data aggregation mechanisms for wireless sensor networks. We first define two specific confidentiality requirements: the sink should first ensure that sensors not participating to the aggregation mechanism do not access the content of the aggregated data (generic confidentiality); moreover, sensors participating to the aggregation mechanism should not access the already aggregated data without the authorization of the sink (end-to-end confidentiality). We show that the use of homomorphic encryption algorithms is essential for aggregation mechanisms and propose the use of an extension of CTRM encryption schemes. In order to protect aggregation mechanisms against node compromise, we first define a key attribution algorithm whereby sensors store several keys with respect to their location in the tree. We then describe a layered secure aggregation mechanism where sensors basically add and suppress some encryption layers with respect to their keying material. We show that this new framework provides both generic and end-to-end confidentiality and is robust against bogus message injections and message losses.

Future work should focus on investigating the problem of key pre-distribution mechanism [**?**] related to the key attribution algorithm that should be self-organized and efficient. We sould also investigate on new solutions that prevent bogus injection rather than minimizing the impact of such attacks.

References