A Secure Aggregation Protocol for Cluster-Based Wireless Sensor Networks With no Requirements for Trusted Aggregator Nodes

Chakib Bekara and Maryline Laurent-Maknavicius *
Institut National des Télécommunications d'Evry
Département Logiciel-Réseaux
9 rue Charles Fourier, 91000 Evry Cedex, France
Email: {Chakib.Bekara, Maryline.Maknavicius}@int-edu.eu

Abstract

Data aggregation is an important feature in wireless sensor networks (WSN), where data inputs at an aggregator node are merged to produce compact small output data, thus reducing the network's transmission overhead and preserving the scarce-battery resources of sensors. Unfortunately, if not secured against attackers, data aggregation can produce erroneous results, which can induce the network operator to take wrong decisions. In this paper, we present a secure aggregation protocol for cluster-based WSN, which is adapted to a large variety of aggregation functions, and which does not rely on trusted aggregator nodes. The inherent transmission overhead is even smaller than in other secure aggregation protocols.

1 Introduction

A WSN is composed of hundreds or thousands of tiny resource-constrained devices, equipped with non-rechargeable and non-replaceable batteries [1]. For such sensors, transmitting is much more energy consuming than computing [2] [3], so the amount of transmitted data on the network must be kept as low as possible, in order to extend the lifetime of the network.

In a typical WSN, data converge from sensors to the BS through multi-hop communications [4]. As a consequence forwarding sensors, especially those surrounding the BS, are rapidly exhausting their energy in forwarding data to the BS. In addition, sensors in the same area, are likely to report the same observed phenomena evolution (i.e. temperature, pressure, etc.) to the BS, thus resulting in redundant information being uselessly routed to the BS, and useless forwarding nodes' batteries depletion. Moreover, in some applications, rather than collecting data at their brut state, the BS is more interested having a synthesized information about a defined region, like the max, min or average of the temperature over the region. To meet all these requirements, several aggregation protocols were introduced [5] [6] [7] [8], where the main objective is to reduce the transmission overhead in the network, using in-network processing, in order to extend the lifetime of the network. Aggregation can be seen as the process by which data, during their forwarding from sensors to the BS, are little-by-little merged by sensors called aggregator nodes, to produce smaller output data. The aggregation processing varies from the simple elimination of duplicated packets, to the compression of data to smaller size, or mathematical operations over sensed data, like sum, average, min, max, etc. However, if the aggregation process is not secured, it can be an easy target to attackers. For instance, an attacker can inject false data or modify transmitted data, or more dangerously compromise or claim to be an aggregator node, in order to significantly falsify the result of aggregation. The main objective of attacking aggregation process, is to produce false aggregation results, and make the BS or the network operator accept false aggregation results, so the wrong decisions and actions are taken.

Several secure aggregation protocols were proposed in the literature [9] [10] [11]. However, these protocols either introduce some heavy communication or computation overheads [11], handle a special kind of aggregation (agreement on the same value) [11], provide a limited resilience against aggregator nodes compromising [9], or require expensive interactive verifications between the BS and sensors [10].

In this paper, we present a new secure aggregation protocol for cluster-based WSN, which does not require trusted aggregator nodes. Our protocol is resilient to nodes compromising including aggregator nodes, and introduces an acceptable communication and transmission overheads. Our protocol allows the BS to verify the authenticity and the validity of the aggregation results, even if all aggrega-

*The length of this article was agreed by the conference Chair.

tor nodes and part of the sensors are compromised in the network.

The rest of the paper is organized as follows. In section 2 we review some secure aggregation protocols with their performances. Section 3 presents our network model, assumptions, security goal and defines our attacker model. Section 4 details our secure aggregation protocol. Section 5 gives a detailed security analysis of our protocol and section 6 its communication overhead. Section 7 compares our protocol with some other protocols in the literature, and section 8 gives an improvement of our protocol. Section 9 gives the limits of our protocol, and section 10 concludes our work.

2 Related works

2.1 Zhu et al. protocol

In [11], *Zhu et al.* present an interleaved hop-by-hop authentication protocol, that achieves secure aggregation even in the presence of at most *k-1* compromised nodes. In the network, there are two kinds of nodes: sensor nodes and forwarding nodes. Sensor nodes form a cluster of at least *k* nodes and monitor an event of interest happening in an area. The aggregator node is the cluster head. Forwarding nodes constitute a path from the aggregator node (cluster head) to the BS, and do no sensing activity. When triggered by the event, the aggregator node sends to the BS a message containing the event's value, if and only if at least *k* nodes in the cluster agree on the same value of the event. To secure the aggregation result, each node in the network establishes upper or/and lower associations with other nodes (resp. upper associate or/and lower associate), depending on its position, as illustrated in Fig. 1.

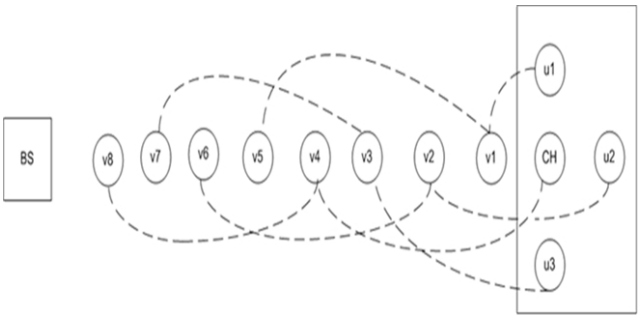

Figure 1. The created upper/lower associations in Zhu et al protocol

Each association represents a secret shared pair-wise key, initially established during network setup phase using some key-establishment process like [12] or [13]. Each node in the network is initially loaded with a secret key it shares with the BS. To send a data "D" to the BS, each node in the cluster (including the aggregator node), generates two different MACs (Message Authentication Code) over "D", one using the secret key it shares with the BS, and the other one using the secret pair-wise key it established with its upper associate. Then, it sends the data along with the two generated MACs to the aggregator node. The aggregator node first verifies that at least *k* nodes agree on the same value "D", and then sends to the BS a message containing the resulted aggregate value "D", the set of MACs generated using upper associations secret pair-wise keys, and an XOR of the *k* MACs generated using the secret keys shared between the BS and the nodes of the cluster. A node *x* on the path from the cluster to the BS can achieve early detection of data modification or injection attempt (either from nodes of cluster or previous nodes in the path). Node *x* verifies that the MAC generated by its lower associate *y* matches the data "D" contained in the message. After that, node *x* replaces node *y*'s MAC by a new MAC it generates using the secret pair-wise key it established with its upper associate. This process is repeated until the message reaches the BS. Once the message reaches the BS, the BS computes the set of *k* MACs over the value "D" using the set of secret keys it shares with the corresponding nodes of the cluster, and then computes an XOR over the generated MACs and compares the generated XOR value with the XOR value contained in the message, to have the insurance that "D" is the originally generated value from the cluster.

Zhu et al. protocol treats only one limited aggregation case (majority *k*-voted value), which severely limits its use in general WSN scenarios. The message sent from the aggregator node to the BS contains at least, *k+1* MACs plus a reading "D", where we suppose that each MAC and reading is 8-byte length, and the length of the path is at least *k* nodes. Depending on where the compromised nodes are placed in the network several nodes in the path are susceptible to forward at least $8 \times (k + 2)$ bytes. In addition, in case no attack is made, most nodes of the path will forward uselessly $8 \times (k + 2)$ bytes, so will deplete early their energy. The *k* value must be sufficiently great to prevent that an attacker can take the control of the cluster.

2.2 Hu et al. protocol

In [9], *Hu et al.* present a secure aggregation protocol, that is resilient to single device key compromise. *Hu et al.* protocol supposes that the network self-organizes on a routing tree where internal nodes, including the root node, are responsible of aggregation, and only leaf nodes are responsible of sensing activities (see Fig. 2). The protocol evolves following two phases: delayed aggregation and delayed authentication. In delayed aggregation, the aggregation of

data sent by nodes of level *k* of the tree is not achieved by nodes of the upper level *k-1*, but is achieved by nodes of level *k-2* (grand-father), where level 0 is the root of tree. In delayed authentication, once the BS receives the partial aggregation results from the root of the tree, it discloses the previously used authentication keys, to enable the aggregator nodes to authenticate the messages they received from their children, and so detect and report any cheating aggregator to the BS, which will invalidate the received aggregation results.

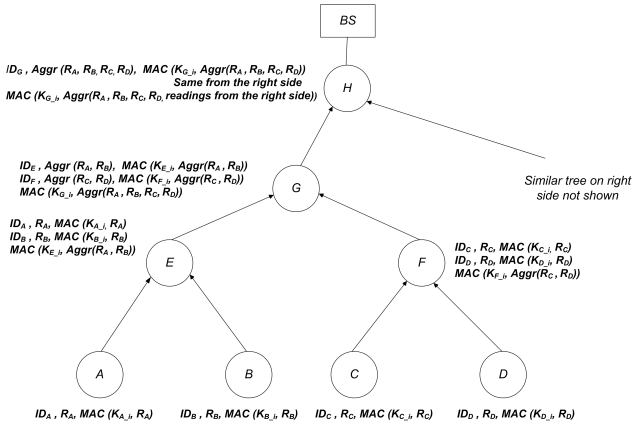

Figure 2. Hu et al tree-based aggregation protocol

Each node *u* in the network is initially loaded with a unique secret key K_u it shares with the BS. From K_u, node *u* derives authentication keys it uses for authenticating the messages it sent to its parent node during the aggregation process. Each node sends one message in each aggregation round. For a node *u*, the i^{th} sent message is authenticated with key $K_{u\_i} = E(K_u, i)$ known only to *u* and the BS, where *E* is an encryption function. The aggregation works as follow:

- Suppose we have a network of *N* nodes, and a binary routing tree of depth $\lg_2 N$, where all leaf nodes are in the same level. In the i^{th} aggregation round, each leaf node *u* (of level $\lg_2 N$) sends a message to its parent containing its identifier, its reading R_u and a MAC generated over the message using key K_{u_i}.

- Each internal node *v* of level $\lg_2 N - 1$, stores the two received messages from its two children, computes using its key K_{v_i} a MAC over the aggregation of the reading of its two children, and then sends a message to its parent containing the identifier, the reading and the MAC generated by each child, in addition to the MAC it computes. Note that node *v* doesn't send the result of aggregation of its children readings.

- Each internal node *x* of level $k < \lg_2 N - 1$, receiving messages from its two children of level *k+1*, stores them, and computes separately the aggregation of the data generated by its left grand-children and its right grand-children of level *k+2*, and a total MAC using its key K_{x_i} generated over the aggregation of data generated by all its grand-children. After that, node *x* sends a message to its parent containing the identifiers of its two children, the two aggregated values it computes and their corresponding received MACs, and the total computed MAC it generates.

- Upon receiving the aggregate results from the tree root, the BS discloses the used authentication keys K_{w_i} of each node *w* on the network. The disclosed keys are authenticated using the μTesla protocol [3]. Each aggregator node will retrieve the keys used by its children and grand-children.

- Each aggregator node checks the validity of the MACs of its children and grand-children, received in the stored messages sent by its children. If a node detects a forged MAC, either from its children or grand-children, it notifies the BS to invalidate the final result of aggregation.

Hu et al. protocol can be used for any aggregation algorithm. The protocol is resilient to aggregator nodes compromising, as long as there is no two consecutive colluding compromised aggregator nodes in the tree. However, this condition can not be guaranteed, because an attacker can compromise nodes in a chosen order in the network. The protocol introduces a heavy communication overhead both during the aggregation phase, and during the key disclosure phase. Indeed, if we take as a simple aggregation function the min of sensors' readings, each internal node in the aggregation tree, sends to its parent a message of 48-byte length, if we consider that each node's identifier is 4-byte length, and each reading and generated MAC is 8-byte length. In addition, the BS widely discloses *N* keys in the network after each aggregation round, where each key is 8-byte length, and nodes of the aggregation tree must forward the keys in reverse path, thus resulting on an additional highly energy consumption.

3 Assumption, network model, adversary model and security objective

3.1 Assumptions and network model

First, we suppose that the BS is a widely trusted and battery unlimited entity, which can not be compromised.

Second, we assume that once all nodes of the network are deployed, they remain static, and they self-organize into clusters to save energy when disseminating data from the BS, and when sending data back to the BS. Different cluster-based routing protocols were proposed in the literature [14] [15] [16] [17], where sensors self-organize into clusters, and where routing and aggregation functions are performed by the cluster-heads (CHs). Our protocol uses *Sun et al.* protocol [17] as the underlying cluster formation protocol, where the formed clusters form disjoint cliques, and inside each cluster (clique) each node is in the communication range of the remaining nodes of the cluster. Consequently, nodes of the same cluster can directly communicate, using one-hop communications only. Once clusters are formed, nodes inside each cluster elect one of them to act as a CH. each CH sends to the BS the list of sensors of its cluster.

Third, we suppose that all nodes can directly reach the BS as supposed in LEACH protocol [14]. In addition, and in order to minimize the communication overhead on the network, only CHs communicate directly with the BS, the remaining nodes communicate only with the nodes of their corresponding cluster. Nodes use two levels of communication power, a minimum power P_{min} when communicating between them inside the same cluster, and at most some higher power $P_{max} > P_{min}$ when a CH communicates with the BS. Of course, a CH near the BS needs less energy to communicate with the BS than a CH far from the BS. As a consequence, nodes define two communication ranges, one small communication range (i.e. 15 or 20 meters) for determining their one-hop neighbors, and one great communication range (i.e. 150 meters) for communications with the BS. To extend the lifetime of network, and to balance the transmission overhead amongst nodes of a cluster, the CH of each cluster is periodically changed. We'll see in section 8, how this assumption can be relaxed, so that not all nodes of the network need to directly reach the BS, but instead, CHs which are far from the BS, rely on other CHs to forward their data to the BS.

Fourth, like the LEACH protocol, we suppose that nodes of a cluster periodically report their readings, using a TDMA scheduling established by the CH after clusters were formed. However, we use a slightly different definition of TDMA scheduling, where the CH divides the time into frames, and during each frame each node of the cluster has two reserved non-consecutive slots: a broadcast slot and a unicast slot. In the broadcast slot a node broadcasts its reading in the cluster, and in the unicast slot it sends a unicast message to the CH. All the broadcast slots are used before the unicast slots can be used. Each node in the cluster remains active during the broadcast slots in order to listen to the broadcast messages of its neighbors of the same cluster, and once it uses it unicast slot, it can go in sleep mode. Readings of sensors inside a cluster can have different values, and the applied aggregation function can be the sum, average, min, max of the readings, or any other mathematically computable function. Each CH sends the result of aggregation of its cluster to the BS, which collects the aggregate values of all clusters, and compute the total aggregate value.

Fifth, we suppose that each node shares a secret key with the BS, initially loaded before deployment. In addition, we suppose that sensors use some key establishment mechanisms, such as [12] [13] for establishing secret pair-wise keys with their neighbors. To do so, each sensor is initially loaded before its deployment with some secret key materials, like a secret polynomial share [12] or a secret line of a secret matrix [13].

3.2 Adversary model and security objective

We assume that an adversary can compromise a portion of sensors of the network including all aggregator nodes (cluster-heads), but can not compromise the BS. The objective of an attacker is to falsify the result of the aggregation output generated by each cluster, and to make the BS accepting false aggregation results. The easiest way for an attacker to achieve this attack, would be compromising the aggregator node and then generating an arbitrary result. The other more complex solution would be compromising a significant portion of sensors of a cluster in order to generate a sufficient amount of bogus readings. As a consequence, that's obvious that aggregator nodes are more attractive for compromising than ordinary nodes. Our main security objective is to protect the aggregation processing against the compromising of aggregators, since aggregator nodes are the basis cornerstone of the aggregation process, and thus represent an ideal target for attackers to falsify the result of aggregation with the minimum effort. Our protocol does not cope with the detection of false readings reported by non-detected compromised nodes, otherwise, this would require some extra protection mechanisms like monitoring nodes behavior, or a majority-based voting mechanism like in [11].

Our protocol ensures the BS that a resulted aggregate value was computed over the original data generated by authorized well-behaving sensors of a cluster, even in the presence of compromised aggregator nodes. So, any attempt of a compromised aggregator node to falsify the result of aggregation, either by modifying readings of well-behaving nodes, or discarding some of them will be detected at the BS.

Table 1. Our notations

Notation	Significance
u, v	Two sensors of the WSN
CH	A cluster-head
C_{CH}	A cluster headed by a cluster-head CH
k	The average size of a cluster
Id_u	4-byte unique identifier of node *u* in the network
$K_{BS,u}$	8-byte secret key shared between node u and the BS.
C_u	A counter shared between the BS and *u* incremented after each aggregation round
$K_{u,v}$	8-byte secret pair-wise key established between *u* and *v*
$\{K_u^n\}$	A one way key-chain of length $n+1$ elements generated by node *u*
K_u^i	The i^{th} key on the key-chain of node *u* where $K_u^{i-1} = H(K_u^i)$, *i=1...n*
K_u^0	The commitment key of the key-chain generated by *u*
R_u	8-byte reading of node *u*
$MAC_K(M)$	8-byte message authentication code generated over *M* using the secret key *K*
H	A one way hash function, with an output length of 8 bytes
$a\|\|b$	a concatenated to b

3.3 Notations

For clarity, the symbols and notations used throughout the paper are listed in Table 1.

4 A secure aggregation protocol for cluster-based WSN

As specified in 3.1, our protocol uses *Sun et al.* protocol as the underlying cluster (cliques) formation protocol. Further details on how clusters are formed are available in [17]. Next a naive secure aggregation protocol for LEACH protocol [14] is presented, followed by the presentation of our secure aggregation protocol.

4.1 A naive secure aggregation protocol

In traditional cluster-based routing protocols, like LEACH [14], TEEN [15] and APTEEN [16], nodes within a cluster send their readings to their CH. The CH applies an aggregation function over the readings to produce an aggregate value, and then sends the aggregate value directly to the BS [14] or to the higher level CH [15] [16]. In such schemes, even if communications between nodes and their CH, and between CH and the BS are secured (encrypted and authenticated), compromising a CH is sufficient to produce false aggregation results, and the BS (or the higher level CH) has no way to detect or verify the misbehavior of the CH, because it has no way to compare the real generated readings on a cluster, with the resulted aggregate value computed by the corresponding CH.

In the case of LEACH protocol [14], a naive secure aggregation protocol can be described as follows:

During an aggregation round, each node *v* of a cluster sends to its CH a message containing its reading, a first MAC computed over its reading and the current counter value C_v using $K_{BS,v}$, and a second MAC computed over the whole message using $K_{CH,v}$:

$$v \rightarrow CH : \underbrace{R_v \| MAC_{K_{BS,v}}(R_v, C_v)}_{1} \| MAC_{K_{CH,v}}(1)$$

Including C_v in the MAC computation protects the BS from replay attacks. The CH can also protect itself against replay attacks, by requiring that the second MAC is computed over the sequence number of each packet sent by *v*.

Upon receiving messages from nodes of its cluster and verifying their authenticity, the CH first generates an XOR-ed MAC over the first MAC ($MAC_{K_{BS,v}}(R_v, C_v)$) received in messages from sensor nodes. Then the CH sends a message to the BS containing the readings of the sensors of the cluster, the computed XOR-ed MAC, and the MAC computed over the entire message using $K_{BS,CH}$:

$$CH \rightarrow BS : \overbrace{\{R_u\}_{u \in C_{CH}} \| \bigoplus_{u \in C_{CH}} MAC_{K_{BS,u}}(R_u, C_u)}^{2} \| MAC_{K_{BS,CH}}(2)$$

In case some nodes fail to send their readings, the CH include their identifiers in the sent message.

This solution is secure, because the BS can easily verify if the CH was honest or not, by first computing the MAC over each reading using the appropriate secret key it shares with the node having sent the reading (assuming the BS knows all nodes of clusters), and then calculates an XOR-ed MAC over the computed MACs and verifies the result with the XOR-ed MAC contained in the message sent by the CH.

This solution is highly energy consuming for the CH. If we suppose that each cluster contains *k* nodes, and that an authenticated packet (network layer) contains a data payload of at most 24 bytes, a header of 12 bytes (source and destination addresses, plus a sequence number), and a generated MAC of 8 bytes computed over the packet, the following transmission overhead applies to a cluster for each aggregation operation:

- Each node sends one packet of 12-byte payload to its CH.

- The CH sends $8\times(k+1)$ bytes to the BS. However, because the transmitted message is unlikely to fit into one packet, the CH splits it into several packets. Consequently, the CH needs to transmit at least $\frac{(k+1)}{3}$ packets, where each frame has a 24-byte payload.

Because transmission is very energy consuming, and because the BS is generally pretty far from sensors, a CH is early depleting its energy.

4.2 Our aggregation protocol

4.2.1 Initialization

Initially, when nodes are deployed, each node establishes pair-wise keys with its one-hop neighbors using some key establishment mechanisms like [12] and [13]. A pair-wise key K_{uv} is used for authenticating any exchanged packet between *u* and *v*. In addition, each node *u* generates a one-way key chain $\{K_u^n\}$ [3] to authenticate its locally broadcasted messages, and sends the commitment key of the key-chain K_u^0 to each neighbor, authenticated with the already established pair-wise key. Then, nodes self organize into clusters (disjoint cliques) according to the protocol described in [17].

Once clusters are formed, nodes inside each cluster elect one of them to act as the cluster-head (CH). Each CH sends to the BS a message containing the list of sensors in its cluster. The CH authenticates the message using the secret key it shares with the BS. Note that in LEACH protocol, CHs are first elected and then clusters are formed, whereas in our protocol, clusters are first formed, and then CHs are elected. As a consequence, in LEACH protocol the maximum distance between two nodes inside a cluster is two-hop, whereas in our protocol the maximum distance between any two nodes of a cluster is exactly one-hop. As such, in our protocol, a new CH selection does not change the cluster sensor members, so there is no extra overhead. In LEACH protocol, most of the time, a new CH selection implies forming new clusters, so an extra communication overhead is induced.

4.2.2 Aggregation protection

In our protocol, no trust is supposed in CHs, which play the role of aggregator nodes. We adopt a slightly different aggregation approach than classical aggregation protocols. Instead of computing and authenticating the aggregation result by the CH only, all nodes of a cluster participate to those procedures. The BS that knows the list of sensors per cluster, can easily check whether the aggregation result of a cluster was approved by the cluster members or not.

In our protocol, the i^{th} aggregation round is done as follows:

1. Each node *u* of a cluster, including the CH, broadcasts its reading R_u to the nodes of its cluster, authenticated with the current key K_u^j of its key chain:

   $$u \rightarrow * : R_u \| MAC_{K_u^j}(R_u) \| K_u^j$$

   Because of only one-hop communications inside a cluster, the time needed for an attacker to intercept the message, and then modify the reading value R_u and generate a new MAC value using the disclosed key K_u^j, is greater than the time needed for the message to reach all nodes of the cluster. Each key is used only once for authentication, each node of the cluster will only accept the first message authenticated with K_u^j, and thus reject any further messages authenticated with K_u^j.

2. Each node v of the cluster, including the CH, receives all the broadcasted messages. For each received message, node *v* first authenticates the disclosed key K_u^j using the stored authenticated key (previously disclosed key) K_u^{j-1}, by checking that $K_u^{j-1} = H(K_u^j)$. Second, it verifies that the received MAC matches the message. If so, it accepts the message and replaces the stored key with the new disclosed one. The previously stored key will be no longer used. After collecting all readings from the cluster, each node locally applies the aggregation function over the readings to produce the resulted aggregate value. If we suppose the aggregation function is the sum of readings, each node *v* computes:

   $$AGR_v = \sum_{u \in C_{CH}} R_u$$

   Then, each node *v* of the cluster calculates a MAC over the concatenation of the computed aggregate result and the current counter value C_v, using $K_{BS,v}$. Also it computes a MAC over the message using $K_{CH,v}$, and sends the following message to the CH:

   $$v \rightarrow CH : \overbrace{AGR_v \| MAC_{K_{BS,v}}(AGR_v, C_v)}^{3}$$
   $$\| MAC_{K_{CH,v}}(3)$$

   Including C_v into the MAC computation, protects the BS from replay attacks. The CH can also self-protect against replay attacks, by requiring that the second MAC being computed over the sequence number of each packet sent from a node of the cluster to the CH

3. The CH verifies the received messages, using the secret pair-wise keys established with nodes of the cluster. Classically, all nodes must report the same aggregate value *AGR*, because all nodes of the cluster view

the same broadcasted messages. Finally, the CH computes an XOR-ed MAC over the MACs generated by nodes of the cluster over the resulted aggregate values, and sends the following message to the BS:

$$CH \rightarrow BS : \overbrace{AGR \| \bigoplus_{v \in C_{CH}} MAC_{K_{BS,v}}(AGR_v, C_v)}^{4}$$

$$\| MAC_{K_{BS,CH}}(4)$$

If a node *v* of the cluster fails to send its computed aggregate value AGR_v, the CH includes Id_v in the message sent to the BS, to notify that the computed XOR-ed MAC was not computed over the contribution of node *v*. In case of conflicting aggregate values, the CH can choose a majority voted aggregate value, and computes the XOR-ed MAC only over the MACs related to the majority voted aggregate value. In this case, the CH must also report the *Id* of each node whose computed aggregate value differs from the majority voted aggregate value.

4. Upon receiving the message sent by a CH, the BS verifies its authenticity using $K_{BS,CH}$. If authenticated, the BS computes a set of MACs over the received aggregate value *AGR*, using the set of secret keys it shares with the nodes of the cluster. The BS then, calculates an XOR-ed MAC over the computed MACs, and then compares the computed XOR-ed MAC with the received XOR-ed MAC. If the two XOR-ed MACs are equal, the BS is ensured that *AGR* value was computed over the original readings generated by the authorized set of sensors on the cluster, otherwise it simply rejects the MAC. It may happen that the received XOR-ed MAC is not computed over all MACs generated by nodes of a cluster, either because some nodes fail to report their result to the CH or some nodes have conflicting aggregate results. Depending on the BS's policy, the BS can accept or deny the received aggregate value. If the BS has defined a threshold parameter *t*, the BS accepts the received aggregate result *AGR*, if and only if the received XOR-ed MAC was computed over at least *t* generated XORs, which means that at least *t* nodes of the cluster must agree on the same aggregate value result.

5 Security analysis of our protocol

As specified in 3.2, our protocol aims to protect the BS from accepting false aggregate results, generated by a compromised, a malicious or a malfunctioning CH. By distributing the task of aggregation over all nodes of a cluster, we alleviate the need to trust a central aggregator node (CH). In our protocol all nodes participate in the computation and the authentication of the resulted aggregate value. As a consequence, a malicious or compromised CH cannot convince the BS of the validity of a false aggregate value it generates, because it cannot compute the MACs of well-behaving non-compromised sensors over the false aggregate value it generates. Depending on the BS's security policy, an attacker has to compromise the entire cluster or part of it in order to make a BS accept a false aggregate result. If the BS requires that the computed aggregate value is computed over all the readings of nodes of a cluster, an attacker must compromise all nodes of the cluster, including the CH, in order that its attack succeeds. If the BS's policy is less strict, it can require that the aggregate value is computed over at least *t* readings generated by *t* sensors of the cluster, where $t < k$. In this case, an attacker must compromise the CH, plus *t-1* sensors of the cluster in order to make its attack possible. In general, the threshold value must be set at least to $t = \frac{k}{2}$ nodes.

Concerning the security of the aggregation process itself, each broadcasted reading R_u is authenticated using node *u*'s current key K_u^i from its key chain. Each key is used only once for authenticating one transmitted reading, and to authenticate the next disclosed key. As a consequence, an attacker cannot masquerade the identity of a non-compromised node by sending readings on behalf of it. The only malicious attack that remains possible is manipulating the readings of compromised nodes.

As stated in 3.2 above, our protocol is not intended to protect the network from malicious readings reported by non-detected compromised nodes or malfunctioning nodes, which can still legitimately authenticate their readings. However, this can be done either by monitoring the readings periodically sent by sensors, or by using a majority voting system. In the monitoring solution, each node monitors the evolution of readings of the nodes of a cluster. If the readings of a node are detected to be significantly different between two successive readings, nodes of a cluster can decide to not take the reading of that node into the computation of the aggregate result. In majority-voting solutions, sensors are assumed densely deployed, so that sensors in each cluster practically report the same value of readings. In this case, the result of aggregation on each cluster is the majority voted value, like in [11]. In this way, each sensor reports as aggregate value the majority voted value.

6 Transmission overhead

The energy dissipation of sensors is mostly due to transmission. As referenced in different works [2] [3], transmitting is more energy consuming than computing, so any proposed protocol for WSN must reduce the transmission

overhead as much as possible. Aggregation protocols were mainly proposed in order to reduce the amount of data transmitted in a WSN, but introducing aggregation security mechanisms has some extra overhead.

Our secure aggregation protocol attempted to introduce small transmission overhead, while providing maximum security level. If we consider the aggregation operation is the sum of sensors readings, and we take the same packet format as used in the secure naive solution, the following overhead applies:

- Each node in the cluster (except the CH) transmits one broadcast packet of 20-byte payload, and one unicast packet (the computed aggregate value) of $16 + \frac{1}{8} \times \lg_2(k)$ byte payload. The two transmitted packets are sent using a minimal power transmission P_{min}.
- The CH transmits one broadcast packet (its reading) of 20-byte payload using a minimal power transmission P_{min} and one unicast packet (the final aggregate value) of $16+\frac{1}{8}\times\lg_2(k)$ byte payload using a maximal power transmission P_{max}.

7 Comparison with previous works

As we can see, our protocol achieves higher security against attacks and nodes compromising (including aggregator nodes) than *Hu et al.* protocol, and is as secure as *Zhu* et al. protocol and the described naive secure aggregation protocol.

7.1 Our protocol vs Zhu et al. protocol

Compared to *Zhu et al.* protocol, our protocol can be used with any computable aggregation function (sum, min, max, average, etc.), while *Zhu et al.* protocol can only be used in majority-voting scenarios. In addition, our protocol introduces less communication overhead on the CHs, where *Zhu et al* protocol introduces heavy communication overhead both on the CHs and on nodes on the path to the BS. However, the transmission overhead of a node inside a cluster is slightly less in *Zhu et al.* protocol (24 bytes) than in our protocol (36 bytes). This is mainly due to the supported aggregation operation in *Zhu et al.* protocol.

Unlike our protocol, *Zhu et al.* protocol allows a cluster which is far away from the BS to reach it through a path, so avoiding a costly long-distance direct communication (e.g. 150 meters), where in our protocol CHs directly communicate with the BS. However, in *Zhu et al.* protocol a CH transmits at least $8 \times (k + 2)$ bytes, where in our protocol a CH transmits only $16 + \frac{1}{8} \times \lg_2(k)$ bytes to the BS, where *k* is the size of the cluster. For a small value of *k=13*, a CH transmits at least 120 bytes in *Zhu et al.* protocol, and only 17 bytes in our protocol. In addition, in *Zhu et al.* protocol, several nodes in the path from the CH to the BS are also involved in the forwarding of the $8 \times (k + 2)$ bytes sent by the CH, so they'll early deplete their batteries. Moreover, in *Zhu et al.* protocol, if the path from the CH to the BS changes, upper/down associations must be updated, resulting in extra communication overhead. In our protocol, no such associations are made.

7.2 Our protocol vs Hu et al. protocol

Our protocol is more secure than *Hu et al.* protocol. Our protocol can provide secure aggregation even if all aggregator nodes were compromised and part of nodes in each cluster. In *Hu et al.* protocol, if at least two successive aggregator nodes *U* and *V* are compromised (where node *U* is the child of *V*), an attacker can easily lie on the real readings generated by sensors on the subtree routed at node *U*. As a consequence, aggregator nodes of higher levels on the tree near the BS became more attractive for attackers, because compromising them can significantly affect the final aggregation result computed at the BS, and with less effort.

Concerning the transmission overhead in *Hu et al.* protocol, each leaf node (sensing node) sends 20 bytes to its parent, where each internal node (aggregator node) sends at least 48 bytes. In addition, when the BS discloses the authentication keys, aggregator nodes participate in broadcasting the $8 \times N$ bytes of keys, where *N* is the network size, and this is very energy demanding for forwarding nodes. Thus, even if the needed transmission power for aggregator nodes is less in *Hu et al.* protocol than in our protocol, the amount of transmitted data sent by a CH (aggregator node) in our protocol is significantly less compared to the amount of transmitted data sent by an aggregator node in *Hu et al.* protocol. Moreover, in *Hu et al.* protocol, half of deployed nodes are aggregator nodes, so half of the nodes will early deplete their energy.

7.3 Our protocol vs naive solution

Our secure aggregation protocol and the naive secure aggregation protocol, are both based on a cluster formation protocol. Our solution is based on *Sun et al.* cliques formation protocol, while the naive solution is based on the well-known LEACH protocol. In the naive solution, CHs spend a lot of energy to send the result of aggregation ($8 \times (k + 1)$ bytes) to the BS, especially when the BS is far from them. In our protocol a CH sends only $16 + \frac{1}{8} \times \lg_2(k)$ bytes as a result of aggregation to the BS. Moreover, in the naive solution, as CH election happens prior to clusters construction, a new CH election results in new formed clusters, and extra communications are necessary for the new CH election and nodes assignment to clusters. In our protocol, clusters are first formed prior to CHs election, so in case of new CHs

being elected, no extra communication overhead is necessary as the cluster remains unchanged. In our protocol, CH nodes can be elected as a CH a longer time than in the naive secure aggregation protocol, where CHs are more solicited for transmission and are likely to suffer more rapidly from battery exhaustion.

Our protocol is more tolerant to aggregator nodes failures than the naive solution. Indeed, in the naive solution, if a CH fails or is unavailable, nodes of the cluster must elect a new CH, and possibly create new cluster(s), so inducing an extra communication overhead. In our protocol, if a CH fails, nodes of a cluster can easily elect one of them to act as a CH, while keeping the same cluster membership. In addition, if a CH failure happens during the aggregation process, our protocol can be adapted to recover from the failure and continue the aggregation from the point of failure, while the naive solution requires the aggregation process to be repeated from the beginning, as a new CH election leads to the creation of new clusters.

8 Improvement of our protocol

In the previous sections, we supposed that each sensor can reach the BS using at most some sufficiently transmission power P_{max}, while using a minimal transmission power $P_{min} < P_{max}$ when communicating with the nodes of its own cluster (clique), like in LEACH protocol. This assumption can be seen as restrictive regarding the deployment area, and highly energy consuming for CHs, especially those far from the BS. We propose here a variation of our protocol, where CHs being far from the BS, rely on other CHs to forward the aggregation results to the BS. In this latter case, CHs being far from the BS consume small transmission power P_{med}, where $P_{min} \leq P_{med} < P_{max}$, in order to communicate with their neighbor cluster-heads. To achieve this, CHs self-organize into multi-hop routing backbone (i.e. hierarchical routing tree or others), where each remote CH can send the result of aggregation of its cluster through a path composed of other CHs. Note that the resulted aggregate value of each remote cluster is sent to the BS, but must not be aggregated as input by the other CHs. While this solution decreases significantly the transmission overhead of CHs which are far from the BS, it increases in the same time the transmission overhead of CHs near the BS, because they need to forward their own aggregate values, but also the aggregate values of remote clusters. However, because the energy consummation for transmission is proportionally related to the square of distance between the sender and receiver [14], and only linearly related to the length of the transmitted message [14], using multihop communications will decrease the overall network's energy consummation.

9 Limitations of our protocol

Our protocol mainly suffers from its restriction to static networks where new incoming nodes are rare (clusters are formed once all nodes are deployed), and where nodes are static once deployed. It's clear that the cost of nodes joining operations is less in the naive secure protocol based on the LEACH cluster formation protocol, than in our protocol based on *Sun et al.* cluster formation protocol. In LEACH protocol, with CH election preceding cluster creation, a new incoming node only requires to broadcast a join message and selects one of the CHs responding to its request as its cluster head. In *Sun et al.* protocol based on nodes organizing themselves into disjoint cliques, we can not ensure that adding a node to any given cluster will result in a clique, so new cliques must be formed again. As a conclusion, initial clusters formation, and nodes adding operations are more costly in *Sun al.* protocol than in LEACH protocol, but periodical CHs election, and recovery operation after CH failures are more costly in LEACH protocol than in *Sun et al.* protocol.

10 Conclusion and perspectives

This paper proposes a new secure aggregation protocol for WSN which does not raise on the usual restrictive condition that the aggregator nodes are trusted nodes, and which introduces little transmission overhead in the network, especially for CHs. Our protocol is resilient to the compromising of all aggregator nodes and part of nodes in the network. Our protocol distributes the task of aggregation inside a cluster, such that an attacker has no way to falsify the result of aggregation other than compromising the aggregator node and a significant portion of nodes in the cluster. Our protocol can be improved to reduce the energy consummation of CHs being far from the BS; the idea is to create a routing backbone based on CHs, that helps preserving the overall network energy. As future works, we plan to adapt our protocol to dynamic WSN, where nodes joining operations might be frequent, and where nodes are moving inside the network.

References

[1] I. F. Akyildiz, W. Su and Y. Sankarasubramaniam, "Wireless sensor networks: a survey", Computer Networks (38), pp. 393-422, 2002

[2] C. Karlof, N. Sastry and D. Wagner."TinySec: A Link Layer Security Architecture for Wireless Sensor Networks". SenSys04, November 35, 2004.

[3] A. Perrig, R. Szewczyk, V. Wen, D. Cullar and J. D. Tygar, "Spins: Security protocols for sensor networks", In

Proc. of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking, pp. 189-199, 2001.

[4] K. Akkaya and M. Younis. "A survey on routing protocols for wireless sensor networks". Ad Hoc Networks 3, pp. 325-349, 2005

[5] B. Krishnamachari, D. Estrin and S. Wicker. "The Impact of Data Aggregation in Wireless Sensor Network". Proceedings of the 22nd International Conference on Distributed Computing Systems, pp. 575- 578, July 2-5, 2002.

[6] C. IntanagonwiI. F. Akyildiz, W. Su and Y. Sankarasubramaniam, "Wireless sensor networks: a survey", Computer Networks (38), pp. 393-422, 2002.

[7] D. Estrin and R. Govindin. "Impact of Network Density on Data Aggregation in Wireless Sensor". Proceedings of the 22 nd International Conference on Distributed Computing Systems, pp. 457- 458, July 2-5, 2002.

[8] J. N. AI-Karaki, R. UI-Mustafa and A. E. Kamal. "Data Aggregation in Wireless Sensor Networks - Exact and Approximate Algorithms". Workshop on High Performance Switching and Routing, pp. 241- 245, April 19-21, 2004.

[9] L. Hu and D. Evans. "Secure Aggregation for Wireless Networks". Proceedings of the 2003 Symposium on Applications and the Internet Workshops, pp. 384- 2003

[10] B. Przydatek, D. Song and A. Perrig. "SIA: Secure Information Aggregation in Sensor Networks". SenSys03, November 5-7, 2003.

[11] S. Zhu, S. Setia, S. Jajodia and P. Ning. "An Interleaved Hop-by-Hop Authentication Scheme for Filtering of Injected False Data in Sensor Networks". Proceedings of the 2004 IEEE Symposium on Security and Privacy, pp. 259-271, May 9-12, 2004.

[12] Yung, "Perfectly secure key distribution for dynamic conferences", In Proc. of the 12th Annual International Cryptology Conference on Advances in Cryptology, Lecture Notes in Computer Science, vol. 17, Springer-verlag, pp. 471-486, 1992.

[13] R. Blom. "An Optimal Class of Symmetric Key Generation". Advances in Cryptography: Proc. of EUROCRYPT 84, Lecture Notes in Computer Science, 209, Springer-Verlag, Berlin, pp. 335-338, 1984.

[14] H. Heinzelman, A. Chandrakasan and H. Balakrishnan. "Energy-efficient communication protocol for wireless microsensor networks". Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, Jan 4-7, 2000.

[15] A. Manjeshwar and D. Agrawal. "TEEN: A protocol for enhanced efficiency in WSN". Proceedings of the 15th International Parallel & Distributed Processing Symposium, pp. 2009-2015, April 23-27, 2001.

[16] A. Manjeshwar and D. Agrawal. "APTEEN: A hybrid protocol for efficient routng and a comprehensive information retrieval in WSN". Proceedings of the International Parallel and Distributed Processing Symposium, pp. 195-202, April 15-19, 2002.

[17] K. Sun, P. Peng, P. Ning and C. Wang. "Secure distributed cluster formation in wireless sensor networks". 22nd Annual Computer Security Applications Conference, December 11-15, 2006

[18] M. Grey and D. Jonhnson. "Computers and intracrbility: A guide to theory of NP-Completeness". W.H Freeman and Company, 1979.