SAPC: A Secure Aggregation Protocol for Cluster-Based Wireless Sensor Networks

Chakib Bekara, Maryline Laurent-Maknavicius, Kheira Bekara

Institut National des Télécommunications d'Evry
9 rue Charles Fourier, 91000 Evry Cedex, France
{chakib.bekara, maryline.maknavicius, kheira.bekara}@int-edu.eu

Abstract. To increase the lifespan of wireless sensor networks (WSN) and preserve the energy of sensors, data aggregation techniques are usually used. Aggregation can be seen as the process by which data sent from sensors to the BS are little-by-little processed by some nodes called aggregator nodes. Aggregators collect data from surrounding nodes and produce a small sized output, thus preventing that all nodes in the network send their data to the BS. Security plays a major role in data aggregation process, especially that aggregators are more attractive for attackers than normal nodes, where compromising few aggregators can significantly affect the final result of aggregation. In this paper, we present SAPC, a secure aggregation protocol for cluster-based WSN, which does not rely on trusted aggregator nodes and thus is immune to aggregators compromising. In addition to security performance, SAPC has a low transmission overhead.

Keywords: WSN, Cluster-based WSN, Data Aggregation, Security

1 Introduction

Sensors on a WSN are generally supplied with non-rechargeable and non replaceable batteries [1]. For such sensors, transmitting is much more energy consuming than computing [2] [3]. For instance, transmitting one bit consumes as much energy as performing one thousand CPU cycle operations [2]. As a consequence, the amount of transmitted data must be reduced, in order to extend the lifetime of the network.

Aggregation techniques are usually used to reduce the transmission overhead in 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 aggregators, to produce smaller output data. The aggregation processing varies from the simple elimination of duplicated data, to the compression of data to smaller size, and mathematical operations over sensed data, like sum, average, min, max, etc. Aggregation aims to reduce the transmission overhead in the network, and consequently to reduce the sensors energy consumption.

Several aggregation protocols were introduced for WSN [5] [6] [7] [8]. However, if the aggregation process is not secured, it can be an easy target for

attackers. For instance, an attacker can inject false data or modify transmitted data, or more dangerously compromise or claim to be an aggregator, 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.

To defeat attacks against aggregation process, 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 [11], provide a limited resilience against aggregator nodes compromising [9], or require expensive interactive verifications between the BS and aggregators [10].

In this paper we present SAPC, 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 transmission overhead. Our protocol allows the BS to verify the authenticity and the validity of the aggregation results, even if all aggregator 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 goals 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. Section 8 gives the limits of our protocol, and section 9 concludes our work.

2 Related Works

2.1 Przydatek et al. Protocol

In [10], Przydatek et al. present a secure information aggregation protocol for WSN. The authors present an aggregate-commitment-prove technique to defeat, with high probability, any attempt of an attacker to falsify the aggregation result by compromising aggregator nodes and part of sensors in the network. In addition, the authors present secure methods for computing the median and the average of measurements, finding the minimum/maximum of measurements, and estimating the size of the network. The protocol guarantees that, in the presence of a malicious aggregator, an accepted aggregation result is within an ε-error bound from the true aggregation result, where epsilon is a verifier parameter. The authors mainly describe their protocol in the presence of one powerful aggregator node in the network. Each sensor in the network shares a secret key with the aggregator node and the network operator which is in a remote location. The aggregation process is done in two steps: aggregation and commitment steps.

In the aggregation step, each sensor sends its authenticated reading to the aggregator. The reading is authenticated with two computed MACs: one gener-

ated using the secret key the sensor shares with the aggregator, and the other one using the secret key the sensor shares with the network operator. Then, the aggregator computes the aggregation result over the n data (measurements) sent by sensors, where n is the size of the network. In the commitment step, the aggregator computes the commitment tree of the sensed data, by computing a binary Merkle hash tree of depth $\log_2 n$. Each leaf of the tree represents the data sent by each node authenticated using the key the node shares with the network operator, the value of each internal node is the hash of the concatenation of its two children nodes, and the root of the tree is the commitment value. The commitment value allows the network operator to verify if the aggregation result was computed over the data generated by the sensors. Finally, the aggregator sends to the network operator the aggregation result along with the commitment value. Note that depending on the aggregation operation, the aggregator will compute one or more commitment trees.

Once the network operator receives the result of aggregation and the commitment value, it initiates an interactive verification phase to verify that the aggregator was honest, and ensures that the aggregation result is close to the true result within ε-error bound. To do that, the network operator requests the aggregator to return $\beta << n$ leaf nodes of the commitment Merkle tree, along with their corresponding path in the tree. For each leaf node, the network operator verifies its authenticity using the secret key it shares with the corresponding node, then it verifies that the corresponding path is consistent by checking if the computed root value is equal to the commitment value. If all the β paths are consistent, the network operator accepts the aggregation result and is ensured, with high probability, that the aggregator is honest.

It's obvious that if there is only one aggregator node in the network, sensors surrounding the aggregator will early deplete their energy on forwarding the readings of sensors that are far from the aggregator. Even if there are several aggregators in the network, assuming that they are powerful is not always true, because in many scenarios aggregators are selected amongst simple sensors which are extremely resource-limited devices. For such sensors, performing interactive verification-proof with the network operator is highly energy consuming, especially if they directly communicate with the network operator using one-hop communications.

2.2 Hu et al. protocol

In [9], Hu et al. present a secure aggregation protocol, that is resilient to a single node compromising. In [9], nodes self-organize in a binary aggregation tree where only internal nodes, including the root node, are responsible of aggregation, while leaf nodes are responsible of sensing activities (see Fig.1). The protocol evolves in two steps: delayed aggregation and delayed authentication. The aggregation process is done in a delayed way, as follows:

1. Each leaf node N sends to its parent a message containing its identifier and its reading. The message is authenticated using a secret key K_N known at

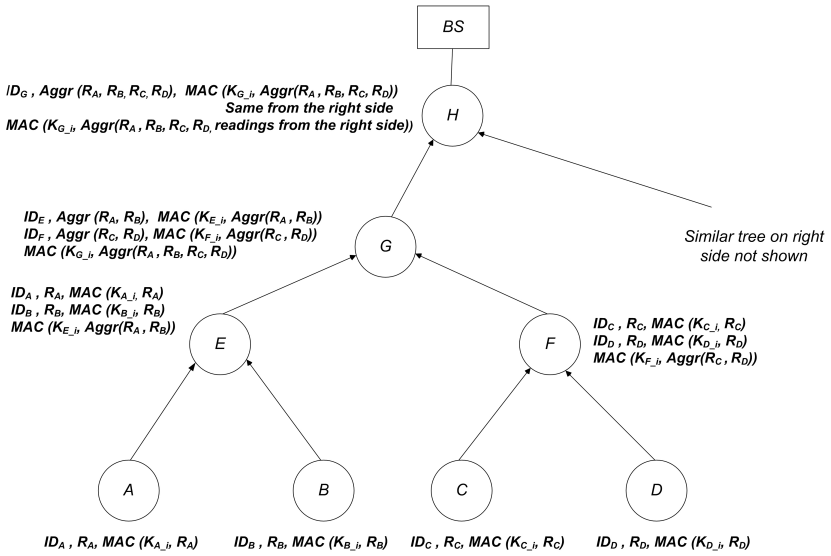

Fig. 1. Hu et al tree-based aggregation protocol

this instant only to N and the BS.

2. Each internal node X of level k in the aggregation tree receives a message from each of its two children Y and Z of level $k+1$ (level 0 is the tree root), and stores the messages. Using the data of the received messages, node X computes the aggregation result of the subtree rooted at its right child Y called AGR_Y, and computes the aggregation result of the subtree routed at its left child Z called AGR_Z. The aggregation result of each subtree is the aggregation of the data generated by the leaf nodes of that subtree. Then, X computes a MAC over the aggregation result of its own subtree ($f(AGR_Y, AGR_Z)$, f being the aggregation function) called MAC_X, using a secret key K_X known to itself and to the BS. Finally, X sends a message to its parent node, containing the two computed partial aggregation values AGR_Y, AGR_Z along with their corresponding MACs MAC_Y and MAC_Z respectively which are contained in the received messages, in addition to the MAC it computes MAC_X. Note that X does not send the aggregation result of its own subtree ($f(AGR_Y, AGR_Z)$), which will be computed by the parent node.

3. The process described in (2) is recursively repeated upstream until reaching the root of the aggregation tree R. In the same way, R computes the aggregation result of its right subtree AGR_{right} and left subtree AGR_{right}, in

addition to a MAC MAC_R over the final aggregation result of the network using a secret key K_R known to itself and the BS. Then, R sends AGR_{right} and AGR_{left} along with their corresponding MACs generated respectively by R's right children and R's left children, in addition to the computed MAC MAC_X to the BS.

4. Upon reception of R's message, the BS discloses all the previously used authentication keys K_W in order to allow each aggregator to authenticate the stored messages (delayed authentication) sent by its children and grandchildren, and thus detect any cheating aggregator node.

The protocol is resilient only to one aggregator node compromising. Indeed, two consecutive compromised nodes in the aggregation hierarchy, can collude to falsify the final aggregation result without being detected in the delayed authentication phase. This leads attackers to target aggregators in the higher hierarchy to significantly disrupt the aggregtion process, and thus produce false aggregation result. The protocol introduces a heavy communication overhead both during the aggregation phase, and during the key disclosure phase. Indeed, during the aggregation phase, data of level k are not aggregated by nodes of level $k-1$, but are aggregated by nodes of level $k-2$, resulting in extra transmission overhead. In addition, during the delayed authentication phase, 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 energy consumption.

3 Background

3.1 Assumptions and network model

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

Second, we assume static WSN, where nodes are immobile once deployed, and where nodes additions are rare.

Third, once nodes are deployed, they self-organize into clusters to save transmission energy. Different cluster formation protocols were proposed in the literature [16] [17] [18] [19], where sensors self-organize into clusters, and where routing and aggregation are performed by the cluster-heads (CHs). Our protocol uses Sun et al. protocol [19] as the underlying cluster formation protocol, where the resulting clusters form disjoint cliques, and inside each cluster (clique) each node is one-hop away from the remaining nodes of the cluster (see Fig.2). Once clusters are formed, nodes inside each cluster elect one of them to act as the CH and as the aggregator. Each CH sends to the BS the list of sensors of its cluster. For routing purpose, we suppose that the set of CHs self-organize into multi-hop routing backbone, so that CHs far from the BS can reach the BS with the minimum spent energy and receive BS's requests. Note that the result

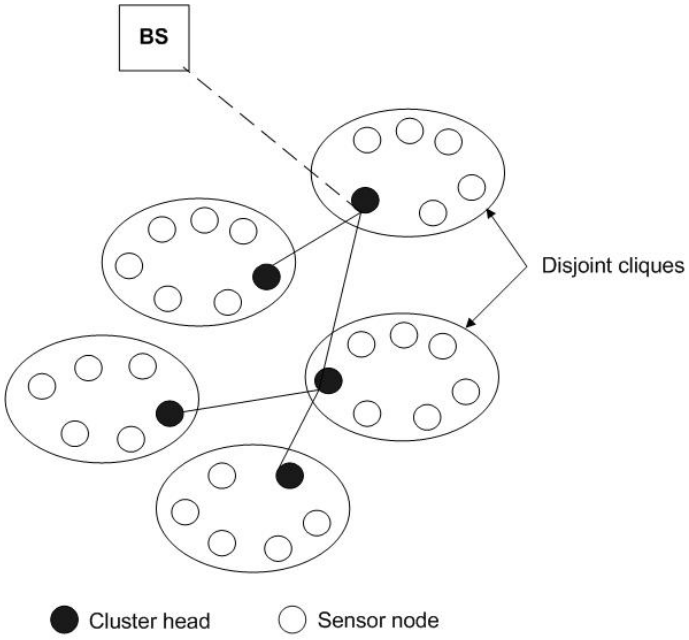

Fig. 2. Our cluster-based WSN

of aggregation of each cluster is sent to the BS, without being aggregated again by other aggregators.

Fourth, we suppose that each node shares a secret key with the BS, initially loaded before deployment. In addition, sensors use some key establishment mechanisms, such as [12] [13] [14] [15] 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], a secret line of a secret matrix [13] or a set of secret keys [14] [15].

3.2 Adversary model and security objective

We assume that an adversary can compromise a (small) portion of sensors of the network including all aggregators (cluster-heads). The objective of an attacker is to falsify the result of aggregation 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 difficult solution would be compromising a significant portion of sensors of a cluster in order to generate a sufficient amount of bogus readings, thus generating an aggregation result which differs from the true one even if the aggregator node was well-behaving. As a consequence, it's obvious that aggregator nodes are more attractive for compromising than ordinary nodes.

Our main security objective is to protect aggregation 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 aggregators. 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.

3.3 Notations

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

Table 1. Our notations

Notation	Significance
u	A sensor node
CH	A cluster-head
C_{CH_i}	A cluster headed by a cluster-head CH_i
k	The average size of a cluster
Id_u	A unique 4-byte identifier of a sensor node u
$K_{BS,u}$	An 8-byte secret key shared between node u and the BS.
C_u	A counter shared between the BS and u to prevent replay attacks
$K_{u,v}$	An 8-byte secret pair-wise key established between nodes 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	An 8-byte reading (measurement) generated by node u
$MAC_K(M)$	An 8-byte message authentication code generated over M using key K
H	A one way hash function, with an output length of 8 bytes
$a\|\|b$	a concatenated to b

4 SAPC: Our proposed secure aggregation protocol

As specified in 3.1, our protocol uses *Sun et al.* protocol as the underlying clusters (cliques) formation protocol. Further details on how clusters are formed are available in [19]. Our protocol evolves in three phases: *initialization*, *cluster*

formation and *aggregation*. The initialization phase occurs before nodes deployment, in which the BS loads each sensor with the necessary secret cryptographic materials. Cluster formation phase occurs when nodes are deployed, in which sensors self-organize into disjoint cliques, and nodes inside each clique elect one of them as the cluster head and aggregator. Aggregation phase occurs after clusters formation, in which the aggregation process is done.

4.1 Initialization phase

The base station loads each node u with a unique identifier Id_u, and a unique secret key $K_{BS,u}$ it shares with it. In addition, it loads u with the necessary secret cryptographic materials, that u uses in-order to establish secret pair-wise keys with its neighbors.

4.2 Cluster formation phase

Initially, when nodes are deployed, each node establishes pair-wise keys with its one-hop neighbors using the loaded cryptographic materials [12] and [13] [14] [15]. A pair-wise key K_{uv} is mainly used for authenticating exchanged packets between u and v, and optionally for encryption purpose. 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. After that, nodes self organize into clusters according to the protocol described in [19].

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. Note that in our protocol clusters are first formed, and then CHs are elected. As a consequence, in our protocol a periodic CH election inside a cluster does not change the cluster sensor members, so there is no extra overhead. In other cluster formation protocols like LEACH [16] TEEN [17] and APTEEN [18], cluster-heads are first elected then clusters are formed centered around the clusters. Thus, a periodic cluster-head election implies new formed clusters, and consequently extra energy consummation due to the exchanged messages.

4.3 Aggregation phase

In our protocol, no trust is supposed in CHs, which play the role of aggregators. To alleviate the need of trusting aggregators, 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, and thus knows whether the aggregator was honest or not. Aggregation process can be done either periodically, or as a response to a

BS request.

In our protocol, the l^{th} aggregation round on a cluster C_{CH_i} is done as follows:

1. Each node $u \in C_{CH_i}$, including CH_i, broadcasts its reading R_u, authenticated with the current key K_u^j of its key chain:

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

2. Each node $v \in C_{CH_i}$, receives all the broadcasted messages. For each received message, node v first authenticates the disclosed key K_u^j using the stored 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. Note that an attacker can not impersonate a node u. Indeed, communications inside a cluster are one-hop only. As a consequence, the time needed for an attacker to intercept a broadcast message sent by u 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, and as such each node of the cluster will only accept the first message authenticated with K_u^j, and thus will reject any further messages authenticated with K_u^j.
   After collecting all readings from the cluster, each node locally applies the aggregation function over the readings to produce the resulted aggregate value: $ARG_v = f(R_u / u \in C_{CH_i})$. If we suppose the aggregation function is the sum of readings, each node $v \in C_{CH_i}$ computes:

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

   Then, each node v computes a MAC over the concatenation of AGR_v and the current counter value C_v, using $K_{BS,v}$. Then, node v sends the following authenticated message to its CH:

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

   Including C_v into the MAC computation, protects the BS from replay attacks. CH_i 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 cluster-head CH_i

3. CH_i verifies the received messages, using the secret pair-wise keys established with nodes of the cluster. Classically, all nodes must report the same hash of the aggregate value, because all nodes of the cluster view the same broadcasted messages, and so compute the same aggregation value. Finally,

CH_i computes an XOR-ed MAC over the MACs generated by nodes of the cluster over the resulted aggregate value, and sends the following message to the BS:

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

The message can be sent directly if the BS is in the transmission range of CH_i, or through a path constituted of other cluster-heads if CH_i is far away from the BS.
If a node $v \in C_{CH_i}$ fails to send its message, CH_i 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 hash aggregate values (so conflicting aggregate values), CH_i can choose a majority voted hash aggregate value, and computes the XOR-ed MAC only over the MACs related to the majority voted hash aggregate value. In this case, CH_i must also report the Id of each node whose computed hash aggregate value differs from the majority voted hash aggregate value.

4. Upon receiving the message sent by CH_i, the BS verifies its authenticity using K_{BS,CH_i}. 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 C_{CH_i}. 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 cluster-head 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 MACs, 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. 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. 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 being computed over at least t readings generated by t sensors of the cluster, where $t < k$, the size of a cluster. 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 bogus 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 aggregation rounds, 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, and thus malicious readings are discarded.

6 Transmission overhead

Aggregation protocols were mainly proposed in order to reduce the amount of data transmitted in a WSN, but securing the aggregation process will certainly have some extra computation and transmission overhead. As referenced in different works [2] [3], transmitting is more energy consuming than computing. As a consequence, any proposed protocol for WSN must introduce the lowest transmission overhead as possible.

Our secure aggregation protocol attempts to introduce a small transmission overhead, while providing maximum security level. If we consider the aggregation operation is the sum of sensors readings, where each reading and cryptographic

key are 8-byte length, and that an authenticated packet 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, the following transmission overhead applies to a cluster for each aggregation operation:

- Each node in the cluster (except the CH) broadcasts in the cluster one packet of 16-byte payload, and sends one unicast packet to its CH (the computed MAC over the aggregate value) of 16 bytes payload.
- The CH broadcasts in the cluster one packet (its reading) of 16-byte payload, and sends one unicast packet (the final aggregate value) to the BS of $16 + \frac{1}{8} \times \lg_2(k)$ byte payload. In addition, each CH which is in the path from a far CH to the BS, will forward one or more packets (aggregation results of distant clusters) of $16 + \frac{1}{8} \times \lg_2(k)$ bytes payload.

7 Comparison with previous works

Our comparison will mainly focus on the resilience to aggregator nodes compromising, and on the energy consumption due to the transmission overhead.

7.1 Resilience to aggregator compromising

Our secure aggregation protocol is resilient to the compromising of all aggregator nodes, and part of sensor nodes compromising depending on the BS's security policy (see Section 5). Hu et al. protocol is resilient to a single aggregator node compromising only, because two consecutive compromised aggregators can collude to produce a false aggregation result, that the BS will accept as valid. Przydatek et al. protocol has a high probability of detecting misbehaving and compromised aggregator nodes that report aggregation results not within the ε-error bound.

7.2 Transmission overhead

In Hu et al. protocol aggregation is done in a delayed way, and half of sensors in the network are aggregators. This implies more energy consummation, because data sent by aggregator nodes of level k in the aggregation tree must be propagated for two-hop before being aggregated by aggregator nodes of level $k-2$. As a consequence, each internal aggregator node will forward the messages sent by its left children and right children, in addition to the MAC it computes. If we consider the aggregation function is the maximum of readings, where each reading and MAC are 8-byte length, each aggregator needs to forward at least 40 bytes. Moreover, during the delayed authentication phase, the BS widely diffuses in the network n cryptographic keys of 8-byte length each, where n is the network size. The keys are forwarded by the aggregator nodes in a reverse path. This result in expensive extra communication overhead.

In Przydatek et al. Protocol, transmission overhead is mainly due to the expensive interactive verification phase, where each aggregator node in the network must provide a proof of the correctness of its aggregation result to the network operator. The proof is a set of β leaf nodes values along with their corresponding path values in the commitment tree, where each value is at least 8-byte length. The length of the path is logarithmic to the number of sensors served by each aggregator. If an aggregator serves 32 sensors, a path in the commitment tree contains 6 hash values, so the length of a path is 48 bytes, and the total amount of data an aggregator sends back to the network operator is $48 \times \beta$ bytes. Depending on the desired aggregation function, we can have different values of β, with β is proportionally related to $\frac{1}{\varepsilon}$ or $\frac{1}{\varepsilon^2}$ and to the size of nodes served by each aggregator. Thus, the smaller is the tolerated error bound ε, the higher is the transmission overhead on the aggregator during the interactive verification phase. Performing the interactive verification directly with the network operator (one-hop communication), seems to be impractical and highly energy consuming especially for those aggregators that are far from it. Even using multi-hop communications remains still costly, because nodes in the path between a remote CH and the network operator will also forward the $48 \times \beta$ bytes.

Our protocol has an acceptable and less transmission overhead, comparing to the two other protocols. First, each sensor node in the cluster locally transmits (using one-hop communication) two packets, of 16-byte payload for each. Each CH transmits also two packets: one broadcast packet of 16-byte payload in the cluster using one-hop communication only, and one packet containing the aggregation result of $16 + \frac{1}{8}\log_2 k$ to the BS using multi-hop communication, and forward other aggregation results $(16+\frac{1}{8}\log_2 k)$ of some other clusters. As consequence, CHs in our protocol have less transmission overhead than in Przydatek et al. Protocol. Comparing to Hu et al. protocol, we must note that in Hu et al. protocol around half sensors of the network are aggregator nodes, while in our protocol based in Sun et al protocol [19], only around $5-7\%$ of sensors are aggregator nodes. As a result, half nodes in Hu et al. protocol will forward at least 40 bytes during aggregation phase, and participate in relaying the $8 \times n$ bytes of the disclosed keys in the delayed authentication phase, while in our protocol few aggregator nodes exist, and each aggregator (CH) forwards its aggregation result and some few other aggregation results of some distant CHs.

8 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. In addition, aggregation in our protocol is only done inside each cluster. The aggregation result of each cluster is sent to the BS, instead of serving as an input to the aggregation process of the next cluster. This seems to be the only way to protect the aggregation process from compromised and misbehaving aggregators. If the aggregation result of a cluster

is aggregated again by the next (upper) aggregator, which is in the path to the BS, the BS has no way to check if the aggregation result of a particular cluster is valid or not. Moreover, the upper aggregator node has no way to check if the aggregation result sent by the down aggregator is valid or not.

9 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.

As a future work, our protocol will be implemented and its performances evaluated through .

Acknowledgement

Authors are thankful to French ANR (Agence Nationale de la Recherche) which funded RNRT project CAPTEUR.

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. C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro and M. 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. Chan, A. Perrig and D. Song. "Random Key Predistibution Schemes for Sensor Networks". In IEEE Symposium on Security and Privacy. Okland California USA, pp. 197-213, 2003.
15. T. Dimitriou T. and I. Krontiris. "A Localized, Distributed Protocol for Secure Information Exchange in Sensor Networks". In Proc. of the 19th IEEE International Parallel and Distributed Processing Symposium, 2005.
16. 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.
17. 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.
18. 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.
19. 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
20. M. Grey and D. Jonhnson. "Computers and intracrbility: A guide to theory of NP-Completeness". W.H Freeman and Company, 1979.