Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2011, Article ID 828414, 19 pages
doi:10.1155/2011/828414

Research Article

Anonymous Aggregator Election and Data Aggregation in Wireless Sensor Networks

Tamás Holczer and Levente Buttyán

Laboratory of Cryptography and Systems Security (CRYSYS), Budapest University of Technology and Economics, Bme-Hit, P.O. Box 91, 1521 Budapest, Hungary

Correspondence should be addressed to Tamás Holczer, holczer@crysys.hu

Received 15 February 2011; Revised 15 May 2011; Accepted 30 June 2011

In mission critical cyber-physical systems, dependability is an important requirement at all layers of the system architecture. In this paper, we propose protocols that increase the dependability of wireless sensor networks, which are potentially useful building blocks in cyber-physical systems. More specifically, we propose two private aggregator node election protocols, a private data aggregation protocol, and a corresponding private query protocol for sensor networks that allow for secure in-network data aggregation by making it difficult for an adversary to identify and then physically disable the designated aggregator nodes. Our advanced protocols resist strong adversaries that can physically compromise some nodes.

1. Introduction

Wireless sensor and actuator networks are potentially useful building blocks for cyber-physical systems. Those systems must typically guarantee high-confidence operation, which induces strong requirements on the dependability of their building blocks, including the wireless sensor and actuator network. Dependability means resistance against both accidental failures and intentional attacks, and it should be addressed at all layers of the network architecture, including the networking protocols and the distributed services built on top of them, as well as the hardware and software architecture of the sensor and actuator nodes themselves. Within this context, in this paper, we focus on the security aspects of aggregator node election and data aggregation protocols in wireless sensor networks.

Data aggregation in wireless sensor networks helps to improve the energy efficiency and the scalability of the network. It is typically combined with some form of clustering. A common scenario is that sensor readings are first collected in each cluster by a designated aggregator node that aggregates the collected data and sends only the result of the aggregation to the base station. In another scenario, the base station may not be present permanently in the network, and the aggregated data must be stored by the designated aggregator node in each cluster temporarily until the base station can eventually fetch the data. In both cases, the amount of communication, and hence, the energy consumption of the network can be greatly reduced by sending aggregated data, instead of individual sensor readings, to the base station.

While data aggregation in wireless sensor networks is clearly advantageous with respect to scalability and efficiency, it introduces some security issues. In particular, the designated aggregator nodes that collect and store aggregated sensor readings and communicate with the base station are attractive targets of physical node destruction and jamming attacks. Indeed, it is a good strategy for an attacker to locate those designated nodes and disable them, because he can prevent the reception of data from the entire cluster served by the disabled node. Even if the aggregator role is changed periodically by some election process, some security issues remain, in particular in the case when the base station is off-line and the aggregator nodes must store the aggregated data temporarily until the base station goes on-line and retrieves them. More specifically, in this case, the attacker can locate and attack the node that was aggregator in a specific time epoch before the base station fetches its stored data, leading to permanent loss of data from the given cluster in the given epoch.

In order to mitigate this problem, we introduced the concept of private aggregator node election, and we proposed the first private aggregator node election protocol in our earlier work [1]. Briefly, our earlier protocol ensures that the identity of the elected aggregator remains hidden from an attacker who observes the execution of the election process. However, our earlier protocol ensures only protection against an external eavesdropper that cannot compromise sensor nodes, and it does not address the problem of identifying the aggregator nodes by means of traffic pattern analysis after the election phase.

In our second paper [2], we addressed the shortcomings of our earlier scheme: we proposed a new private aggregator node election protocol that is resistant even to internal attacks originating from compromised nodes, and we also proposed a new private data aggregation protocol and a new private query protocol which preserved the anonymity of the aggregator nodes during the data aggregation process and when they provide responses to queries of the base station. In our second private aggregator node election protocol, each node decides locally in a probabilistic manner to become an aggregator or not, and then the nodes execute an anonymous veto protocol to verify if at least one node became aggregator. The anonymous veto protocol ensures that nonaggregator nodes learn only that there exists at least one aggregator in the cluster, but they do not learn any information on its identity. Hence, even if such a nonaggregator node is compromised, the attacker learns no useful information regarding the identity of the aggregator.

However, our second protocol used a special broadcast communication scheme, which assumed the existence of a Hamilton cycle in the cluster. Creating a Hamilton cycle in a wireless sensor network is a difficult and energy-consuming problem, therefore in this paper, we relax this assumption and we use spanning trees, which are easy to construct.

Another important improvement in this paper is that we address the problem of misbehaving nodes that try to obfuscate the values received by the operator. We analyze how a malicious node can mislead the operator and propose an algorithm which can detect the presence of misbehaving nodes.

Our protocols can be used to protect sensor network applications that rely on data aggregation in clusters, and where locating and then disabling the designated aggregator nodes is highly undesirable. Such applications include high-confidence cyber-physical systems where sensors and actuators monitor and control the operation of some critical physical infrastructure, such as an energy distribution network, a drinking water supply system, or a chemical pipeline. A common feature of these systems is that they have a large geographical span, and therefore, the sensor network must be organized into clusters and use in-network data aggregation in order to ensure scalability and energy-efficient operation. Moreover, due to the mission critical nature of these applications, it is desirable to prevent the identification of the aggregator nodes in order to limit the impact of a successful attack against the sensor network. Our first protocol that resists only an external eavesdropper is less complex than our second protocol that works in a stronger attacker model. Hence, the first protocol can be used in case of strong resource constraints or when the risk of compromising sensor nodes is limited (e.g., it may be difficult to obtain physical access to the nodes). Our second protocol is needed when the risk of compromised and misbehaving nodes cannot be eliminated by other means.

The remainder of the paper is organized as follows. In Section 2, we introduce our system and attacker models. In Section 3, we present our basic aggregator election protocol which can withstand external attacks, while in Section 4, we introduce our advanced protocols, which can withstand internal aggregator identification and scamming attackers as well. In Section 5, we give an overview of some related work, and in Section 6, we conclude the paper and sketch some future research directions.

2. System and Attacker Models

A sensor network consists of sensor nodes that communicate with each other via wireless channels. Every node can generate sensor readings and store it or forward it to another node. Each node can directly communicate with the nodes within its radio range; those nodes are called the (one-hop) neighbors of the node. In order to communicate with distant nodes (outside the radio range), the nodes use multi-hop communications. The sensor network has an operator as well, who can communicate with some of the nodes through a special node called base station, or can communicate directly with the nodes if the operator moves close to the network.

Throughout the paper, a data-driven sensor network is envisioned, where every sensor node sends its measurement to a data aggregator regularly. Such data driven networks are used for regular inspection of monitored processes notably in critical infrastructures. Event-driven networks can be used for reporting special usually dangerous but infrequent events like fire in a building. There is no need of clustering and data aggregation in event-based systems, thus private cluster aggregator election and data aggregation is not applicable there. The third kind of network is the query-driven network, where the operator sends a query to the network, and the network sends a response. This kind of functionality can be used with data-driven networks, and can have privacy consequences, like the identity of the answering node should remain hidden.

In the following, we will assume, that the time is slotted, and one measurement is sent to the data aggregator in each time slot. The time synchronization between the nodes is not discussed here, but a comprehensive survey can be found in [3].

It is assumed that every node shares some cryptographic credentials with the operator. These credentials are unique for every node, and the operator can store them in a lookup table, or can be generated from a master key and the node's identifier on demand. The exact definition of the credentials can be found in Sections 3.1 and 4.1.

The nodes may be aware of their geographical locations, and they may already be partitioned into well-defined

geographical regions. In this case, these regions are the clusters, and the objective of the aggregator election protocol is to elect an aggregator within each geographical region. We call this approach location-based clustering; an example would be the PANEL protocol [4].

A kind of generalization of the position-based election is the preset case, where the nodes know the cluster ID they belong to before any communication. Here the goal of the election is to elect one node in every preset cluster. This approach is used in [2].

Alternatively, the nodes may be unaware of their locations or cluster IDs and know only their neighbors. In this case, the clusters are not predetermined, but they are dynamically constructed parallel to the election of the aggregators. Basically, any node may announce itself as an aggregator, and the nodes within a certain number of hops on the topology graph may join that node as cluster members. We call this approach topology-based clustering; an example would be the LEACH protocol [5].

The location-based and the topology-based approaches are illustrated in Figure 1.

Both approaches may use controlled flooding of broadcast messages. In case of location-based or preset clustering, the scope of a flood is restricted to a given geographic region or preset cluster. Nodes within that region rebroadcast the message to be flooded when they receive it for the first time. Nodes outside of the region or having different preset cluster IDs simply drop the message. In case of topology-based clustering, we assume that the broadcast messages has a time-to-live field that controls the scope of the flooding. Any node that receives a broadcast message with a positive TTL value for the first time will automatically decrement the TTL value and rebroadcast the message. Duplicates and messages with TTL smaller than or equal to zero are silently discarded. When we say that a node broadcasts a message, we mean such a controlled flooding (either location-based, preset, or topology-based, depending on the context). In Section 4, we will use connected dominating sets (CDSs) to implement efficient broadcast messaging. The concept of CDS will be introduced there.

We call the set of nodes which are (in the location-based and the preset case) or can potentially be (in the topology-based case) in the same cluster as a node S the *cluster peers* of S. Hence, in the location-based case, the cluster peers of S are the nodes that reside within the same geographic region as node S. In the preset case, the cluster peers are the nodes sharing the same cluster ID. In the topology-based case, the set of cluster peers of S usually consists in its n-hop neighborhood, for some parameter n. The nodes may not explicitly know all their cluster peers.

The main functional requirement of any clustering algorithm is that either node S or at least one of the cluster peers of S will be elected as aggregator.

The leader of each cluster is called cluster aggregator, or simply aggregator. In the following we will use aggregator, cluster aggregator, and data aggregator interchangeably.

As mentioned in Section 1, an attacker can gain much more information by attacking an aggregator node than attacking a normal node. To attack a data aggregator node either physically or logically, first the attacker must identify that node. In this paper we assume that the attacker's goal is to identify the aggregator (which means that simply preventing, jamming, or confusing the aggregation is not the goal of the attacker). In Section 4.5 we go a little further, and analyze what happens if a compromised node does not follow the proposed protocols in order to mislead the operator.

An attacker who wants to discover the identity of the aggregators can eavesdrop the communication between any nodes, can actively participate in the communication (by deleting modifying and inserting messages) and can physically compromise some of the nodes. A compromised node is under the full control of the attacker, the attacker can fully review the inner state of that node, and can control the messages sent by that node.

Compromising a node is a much harder challenge for an attacker then simply eavesdropping the communication. It requires physical contact with the node and some advanced knowledge, however it is far from impossible for an attacker with good electrical and laboratory background [6]. So we propose two solutions. The first basic protocol can fully withstand a passive eavesdropper, but a compromising attacker can gain some knowledge about the identities of the cluster aggregators. The second advanced protocol can withstand a compromising attacker as well, with only leaking information about the compromised nodes.

In case of a passive adversary, a rather simple solution could be based on a commonly shared global key. Using that shared global key as a seed of a pseudorandom number generator, every node can construct locally (without any communications) the same pseudo randomly ordered list of all nodes. These lists will be identical for every node because all nodes use the same seed and the same pseudo random number generator. Then, the first A nodes of the list are elected aggregators such that every node can communicate with a cluster aggregator and no subset of A covers the whole system. An illustration of the result of this algorithm can be seen on Figure 1 for location-based and topology-based cluster aggregator election.

The problem with this solution is that it is not robust: compromising a single node would leak the common key, and the adversary could compute the identifier of all cluster aggregators. While we do not want to fully address the problem of compromised nodes in the first protocol, we still aim at a more robust solution than the one described above. In particular, the system should not collapse by compromising just a single or a few nodes.

The second protocol can withstand the compromise of some nodes without the degradation of the privacy of the cluster aggregators. This protocol meets the following goals and has the following limitations.

(i) The identity of the noncompromised cluster aggregators remains secret even in the presence of passive and active attackers or compromised nodes.

(ii) The attacker can learn whether the compromised node is an aggregator.

(iii) An attacker can force a compromised node to be aggregator, but does not know anything about the existence or identity of the other aggregators.

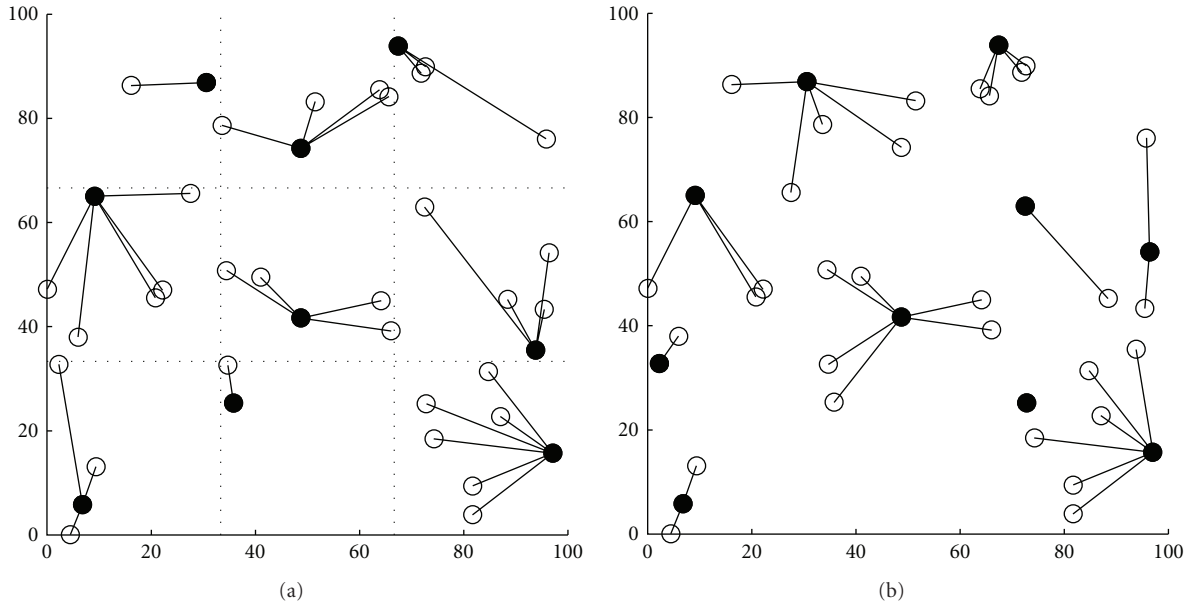

Figure 1: Result of a location-based (a) and topology-based (b) one-hop aggregator election protocol. Solid dots represent the aggregators, and empty circles represent cluster members.

(iv) The attacker cannot achieve that no aggregator is elected in the cluster, however all the elected aggregator(s) may be compromised nodes.

The main difference between the first and second protocol is the following. The first protocol is very simple, but not perfect as a compromised node can reveal the identity of the aggregators. The second protocol requires more complex computations, but offers anonymity in case of node compromise as well. In some cases such complex computations are outside the capabilities of the nodes (or the probability of compromise is low), but anonymity is still required by the system. In these cases we suggest to use the first protocol. If the probability of node compromise is not negligible, then the use of the second protocol is recommended.

3. Basic Protocol

In this section, we describe the basic protocol that we propose for private aggregator node election. An important component of this basic protocol will be presented in Section 3.2, where we also describe how to set the parameters of the protocol. We first give a brief overview of the basic principles of our protocol and present the details later.

3.1. Protocol Description. We assume that the nodes are synchronized (see [3] for a survey on time synchronization mechanism for sensor networks), and each node starts executing the protocol roughly at the same time. The protocol terminates after a predefined fix amount of time. During the execution of the protocol, any node that has not received any aggregator announcement yet may decide to become an aggregator, in which case, it broadcasts an aggregator announcement message announcing itself as a cluster aggregator. This message is broadcast among the cluster peers of the node sending the announcement (see Section 2). Upon reception of a cluster aggregator announcement, any node that has neither announced itself as a cluster aggregator nor received any such announcement yet will consider the sender of the announcement as its cluster aggregator. In order to prevent an external observer to learn the identity of the cluster aggregators, all messages sent in the protocol are encrypted such that only the nodes to whom they are intended can decrypt them. For this, we assume that each node shares a common key with all of its cluster peers (an overview of available key establishment mechanisms for sensor networks can be found in [7]). In addition, in order to avoid that message originators are identified as cluster aggregators, the nodes that will be cluster members are required to send dummy messages that cannot be distinguished from the announcements by the external observer (i.e., they are encrypted and disseminated in the same way as the announcements).

Note that the proposed basic protocol considers only either pairwise keys between the neighboring nodes or group keys shared between sets of neighboring nodes, so no global key is assumed. Such pairwise or group keys can be established by the techniques proposed in [7]. The key establishment can be based on randomly selected key sets. In such a protocol, the probability that neighboring nodes share a common key is high, and the unused keys are deleted [8]. The key establishment can be also based on a common key which is deleted after some short time when the neighbors are discovered [9]. Any node that owns the common key can generate a pairwise key with a node which owns or previously owned the common key. The basic method for exchanging a group/cluster key with the neighboring nodes is to send

```
start T1, expires in rand(0,τ) //timer, expires in round 1
start T2, expires in rand(τ,2τ) //timer, expires in round 2
announFirst = (rand(0,1) ≤ γ)
CAID = −1 // ID of the cluster aggregator of the node
while T1 NOT expired do
   if receive ENC(announcement) AND (CAID = −1) then
      CAID = ID of sender of announcement
   end if
end while
// T1 expired
if announFirst AND (CAID = −1) then
   broadcast ENC(announcement);
   CAID = ID of node itself;
else
   broadcast ENC(dummy);
end if
while T2 NOT expired do
   if receive ENC(announcement) AND (CAID = −1) then
      CAID = ID of sender of announcement
   end if
end while
// T2 expired
if (NOT announFirst) AND (CAID = −1) then
   broadcast ENC(announcement);
   CAID = ID of node itself;
else
   broadcast ENC(dummy);
end if
```

Algorithm 1: Basic private cluster aggregator election algorithm.

the same random key to each neighbor encrypted with the previously exchanged pairwise keys.

The pseudocode of the protocol is given in Algorithm 1, and a more detailed explanation of the protocol's operation is presented below. The protocol consists of two rounds, where the length of each round is τ. The nodes are synchronized; they all know when the first round begins, and what the value of τ is. At the beginning, each node starts two random timers, T1 and T2, where T1 expires in the first round (uniformly at random) and T2 expires in the second round (uniformly at random). Each node also initializes at random a binary variable, called announFirst, that determines in which round the node would like to send a cluster aggregator announcement. The probability that announFirst is set to the first round is γ, which is a system parameter. The setting of γ is elaborated in Section 3.2.

In the first round, every node S waits for its first timer T1 to expire. If S receives an announcement before T1 expires, then the sender of the announcement will be the cluster aggregator of S. When T1 expires, S broadcasts a message as follows: if announFirst is set to the first round and S has not received any announcement yet, then S sends an announcement, in which it announces itself as a cluster aggregator. Otherwise, S sends a dummy message. In both cases, the message is encrypted (denoted by ENC() in the algorithm) such that only the cluster peers of S can decrypt it.

Table 1: Estimated time of the building blocks on a MICAz.

Algorithm	Generation [ms]	Verification [ms]
SHA-1 [10]	1.4	—
RSA 1024 bit [11]	12040	470
RC4 [10]	0.1	0.1
RC5 [10]	0.4	0.4

The second round is similar to the first round. When T2 expires S broadcasts a message as follows: If announFirst is set to the second round and S has not received any announcement yet, then S sends an announcement, otherwise, S sends a dummy message. In both cases, the message is encrypted.

It is easy to see that at the end of the second round each node is either a cluster aggregator or it is associated with a cluster aggregator whose ID is stored in variable CAID. Without the second round, a node can remain unassociated, if it sends and receives only dummy messages in the first round. In addition, a passive observer only sees that every node sends two encrypted messages, one in each round. This makes it difficult for the adversary to identify who the cluster aggregators are (see also more discussion on this in the next section). In addition, if a node is compromised, the adversary learns only the identity of the cluster aggregators whose announcements have been received by the compromised node.

In WSNs, it must be analyzed what happens if some messages are delayed or lost in the noisy unreliable channel. Two cases must be analyzed, dummy messages and announcements. If a dummy message is delayed or not delivered successfully to all recipients, then the result of the protocol is not modified as dummy messages serve for only covering the announcements. If an announcement is delayed or not delivered to a node, then the recipient will not select the sender as cluster aggregator. It will select a node who sent the announcement later or the node elects itself and sends an announcement. The message loss may modify the resulting set of cluster aggregators, but neither harm the anonymity of the elected aggregators, nor harm the original goal of cluster aggregator election (a node must be either a cluster aggregator or a cluster aggregator must be elected from the nodes cluster peers).

Note that two neighboring nodes can send an announcement at the same time with some small probability. Actually, it is not a problem in the protocol. The only result is that both nodes will be cluster aggregators independently. As it is not conflicting with the original goal of cluster aggregator election, this infrequent situation does not need any special attention.

The overhead introduced by the basic protocol is sending two encrypted messages for each election round. Other protocols [4, 5] use one (or zero) unencrypted messages to elect an aggregator. So the number of messages sent in the election phase is slightly larger compared to other solutions. The symmetric encryption also causes some extra overhead (for details, see Table 1, rows with RC4 and RC5).

3.2. Protocol Analysis. In this section, the previously suggested basic protocol is analyzed. As defined in Section 2, the main goal of the attacker is to reveal the identity of the cluster aggregators. To do so, the attacker can eavesdrop modify and delete messages, and can capture some nodes.

First the logical attacks are analyzed where the attacker does not capture any nodes, then the results of a node capture.

The attackers main goal is to reveal the identity of the cluster aggregators. As all the internode communication is encrypted and authenticated, it cannot get any information from the messages themselves, but it can get some side information from simple traffic and topology analysis.

3.2.1. Density-Based Attack. Thanks to the dummy messages and the encryption in the basic protocol, an external observer cannot trivially identify the cluster aggregators; however, it can still use side information and suspect some nodes to be cluster aggregators with higher probability than some other nodes. Such a side information is the number of the cluster peers of the nodes. This number correlates with the local density of the nodes, that is why this attack is called density-based attack. Indeed, the probability of becoming a cluster aggregator depends on the number of the cluster peers of the node. For instance, if a node does not have any cluster peers, it will be a cluster aggregator with probability one. On the other hand, if the node has a larger number of cluster peers, then the probability of receiving an announcement from a cluster peer is large, and hence, the probability that the node itself becomes cluster aggregator is small. Note also that the number of cluster peers can be deduced from the topology of the network, which may be known to the adversary.

The probability of becoming a cluster aggregator is approximately inversely proportional to the number of cluster peers:

$$\Pr(\mathrm{CA}(S)) \cong \frac{1}{D(S)}, \tag{1}$$

where $\mathrm{CA}(S)$ is the event of S being elected cluster aggregator, and $D(S)$ is the number of cluster peers of node S. Figure 2 illustrates this proportionality where the curve belongs to (1) and the plotted dots correspond to simulation results (100 nodes, random deployment, one hop communication, and topology-based clustering).

Two approaches can be used to mitigate this problem. One is to take the number of cluster peers of the nodes into account when generating the random timers for the protocol. The second is to balance the logical network topology in such a way that every node has the same number of cluster peers. In the following a possible solution for both approaches is introduced.

The first approach can be the fine tuning of the distributions. It is not analyzed here deeply. It can only slightly modify the probabilities of being cluster aggregator, so it has no large effects. An example can be seen on Figure 3, where the 10th power of $D(S)$ is used as a normalizing factor, when γ (probability of sending an announcement in the first round) is computed. The coefficients of the polynomial are set as resulting curve is the closest to uniform distribution. It can be seen that modifying γ on a per node basis does not eventually reach its goal; the normalized distribution is far from uniform. Actually by modifying γ, the other attack discussed in the next section can be mitigated, so here we propose a solution which does not set the γ parameter.

The second approach modifies the number of cluster peers of a node to reach a common value. Let us denote this value by α.

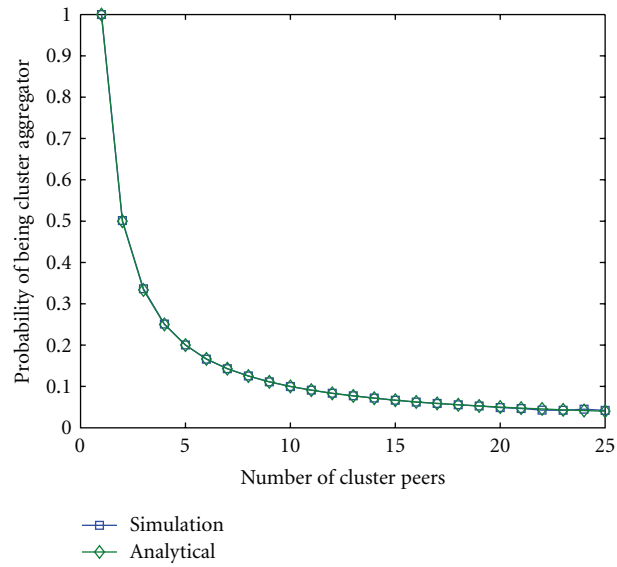

Figure 2: Probability of being cluster aggregator as a function of the number of cluster peers.

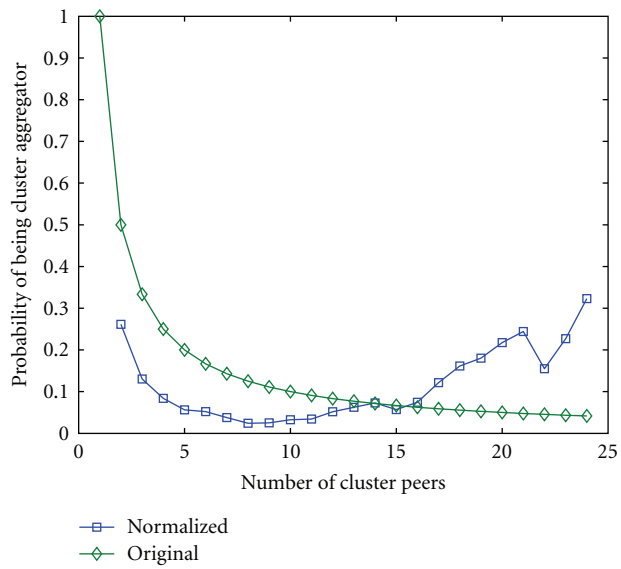

Figure 3: Probability of being cluster aggregator as a function of number of cluster peers. The analytical values comes from (1), while the simulation values come from simulation, where the γ probabilities are normalized with the number of cluster peers of the nodes.

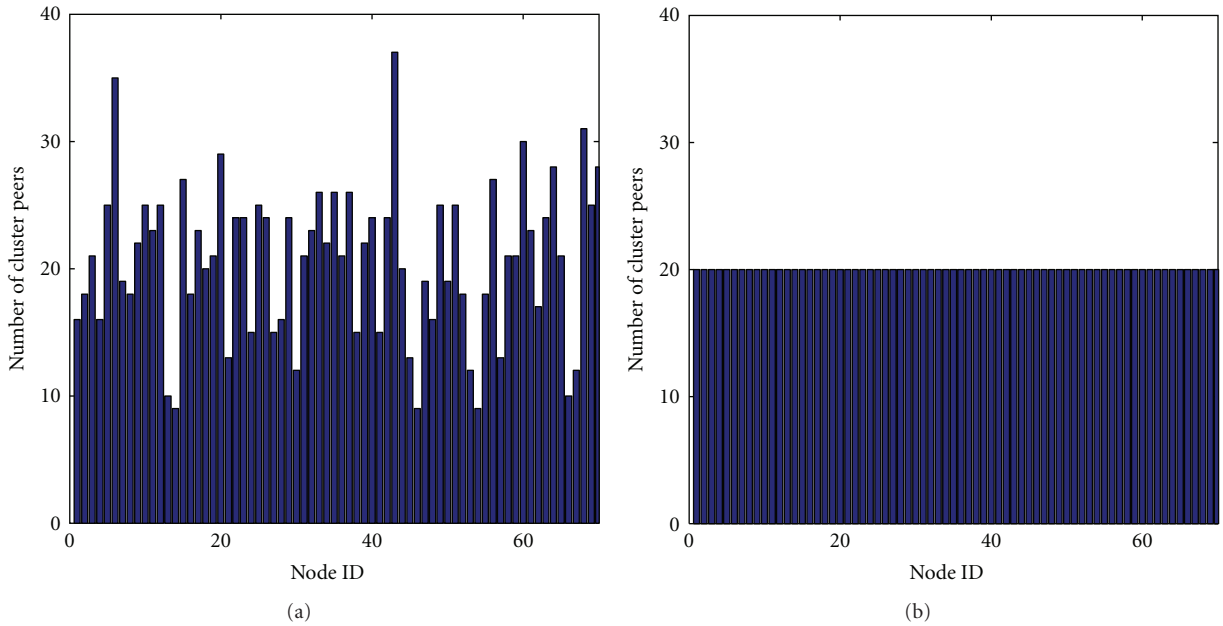

Figure 4: Result of balancing. The 70 nodes are represented on the x-axis. The number of cluster peers before (a), and after (b) the balancing are represented on the y-axis.

An efficient approach to mitigate this problem is to modify the number of cluster peers such that it becomes a common value α for all of them. In theory, this common value can be anything between 1 and the total number N of the nodes in the network. In practice, it should be around the average number of cluster peers, which can be estimated locally by the nodes. For example, assuming one-hop communications (meaning that the cluster peers are the radio neighbors), the following formula can be used:

$$\alpha = (N-1)\frac{R^2\pi}{A} + 1 \simeq E(D(S)), \tag{2}$$

where R is the radio range, and A is the size of the total area of the network. The formula is based on the fact that the number of cluster peers is proportional to the ratio between radio coverage and total area. Similar formulae can be derived for the general case of multi-hop communication.

If a node S has more than α cluster peers it can simply discard the messages from $D(S) - \alpha$ randomly chosen cluster peers. If S has less than α cluster peers it must get new cluster peers by the help of its actual cluster peers (if S has not got any cluster peers originally, then it will always become a cluster aggregator). The new cluster peers can be selected from the set of cluster peers of the original cluster peers. To explore the potential new cluster peers, every node can broadcast its list of cluster peers within its few hop neighborhood before running the basic protocol. From the lists of the received cluster peers, every node can select its $\alpha - D(S)$ new cluster peers uniformly at random. Then, the basic aggregator election protocol can be executed using the balanced set of cluster peers. An example for this balancing is shown in Figure 4 (70 nodes, random deployment, one hop communication, and topology-based clustering).

After running the balancing protocol, every node can approach the envisioned α value. The advantage of the balancing protocol is that however an attacker can gather the information about the number of cluster peers, this number is efficiently balanced after the protocol. The drawback of this solution is that it requires the original cluster peers to relay messages between distant nodes. One can imagine this solution as selectively increasing the TTL of protocol messages creating much larger neighborhoods.

3.2.2. Order-Based Attack. Another important side information an attacker can use is the *order* in which the nodes send messages in the first round of the protocol. Indeed, the sender of the ith message will be cluster aggregator if none of the previous $i-1$ messages are announcements (but dummies) and the ith message is an announcement. Thus, the probability P_i that the sender of the ith message becomes cluster aggregator depends on i and parameter γ:

$$P_i = (1-\gamma)^{i-1}\gamma, \quad 1 \le i \le n. \tag{3}$$

The $(n+1)$th element of the distribution is the probability that no announcement is sent in the first round:

$$P_{n+1} = (1-\gamma)^n \tag{4}$$

in which case the sender of the first message of the second round must be a cluster aggregator.

The entropy of this distribution characterizes the uncertainty of the attacker who wants to identify the cluster aggregator using the order information. Assuming that

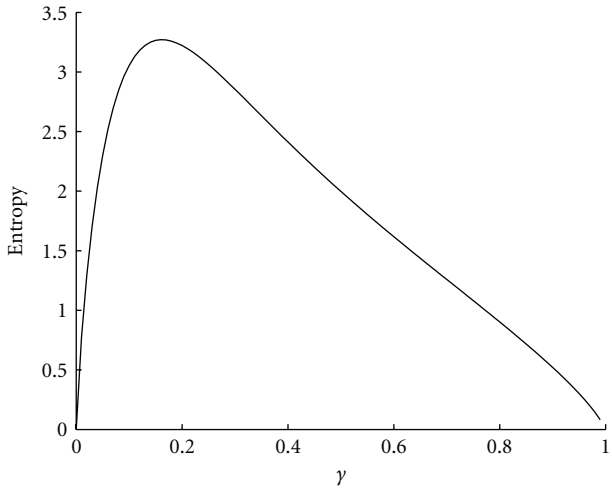

Figure 5: Entropy of the attacker as a function of sending announcement in the first round (γ). Number of nodes in one cluster: 10.

the number of cluster peers has been already balanced, this entropy can be calculated as follows:

$$\begin{aligned} H &= -\sum_{i=1}^{n+1} P_i \log P_i \\ &= -\sum_{i=1}^{n} \left((1-\gamma)^{i-1} \gamma \log\left((1-\gamma)^{i-1} \gamma \right) \right) \\ &\quad - (1-\gamma)^n \log (1-\gamma)^n, \end{aligned} \tag{5}$$

where γ is the probability of sending an announcement in the first round and n is the balanced number of cluster peers.

In Figure 5, we plotted formula (5). If γ is large, then the uncertainty of the attacker is low, because one of the first few senders will become the cluster aggregator with very high probability. If γ is very small then the uncertainty of the attacker is small again, because no cluster aggregator will be elected in the first round with high probability, and therefore, the first sender of the second round will be the cluster aggregator. The ideal γ value corresponds to the maximum entropy, which can be easily computed by the nodes locally from formula (5). For instance, Table 2 shows some ideal γ values for different number of nodes in one cluster. The fifth row ($H_{\max}$) shows the maximal entropy (uncertainty) that any kind of election protocol can achieve with the given number of nodes. This is achieved if every node is equiprobably elected from the viewpoint of the attacker. This value is closely approached by $H(\hat{\gamma})$, where $\hat{\gamma}$ is very close to the optimal solution (the difference between the found value and the optimal value can be arbitrarily small, and depends on the number of iterations the estimation algorithm uses). Using the found $\hat{\gamma}$ value, the order of the messages has no meaning for the attacker.

3.2.3. Node Capture Attacks. If an attacker can compromise a node, it can reveal some sensitive information, even when the system uses the local key-based protocol. If the compromised node is a cluster aggregator, then all the previously stored messages can be revealed. The attacker can decide to demolish the node, modify the stored values, simply use the captured data, or modify the aggregation functions.

If the compromised node is not a cluster aggregator, then the attacker can reveal the cluster aggregator of that node, which can result in the same situation described in the previous paragraph.

Table 2: Optimal γ values ($\hat{\gamma}$) for different number of nodes in one cluster. Achieved entropy ($H(\hat{\gamma})$) and maximal entropy ($H_{\max} = \log_2 n$).

n	10	25	50	100
$\hat{\gamma}$	0.167	0.082	0.049	0.027
$n\hat{\gamma}$	1.67	2.05	2.45	2.7
$H(\hat{\gamma})$	3.281	4.410	5.312	6.218
$H_{\max}$	3.322	4.644	5.644	6.644

3.3. Data Forwarding and Querying. The problem of forwarding the measured data to the aggregators without revealing the identity of the aggregators is a well-known problem in the literature, called anonymous routing [12–14].

Anonymous routing lets us route packets in the network without revealing the destination of the packet. A short overview of anonymous routing can be found in Section 5.

With anonymous routing any node can send the measurements to the aggregators without revealing the identity of it. An operator can query the aggregator with the help of an ordinary node which uses anonymous routing towards the aggregator.

Anonymous routing introduces significant overhead in the traffic. However this can be partially mitigated by synchronizing the data transmissions. Instead of suggesting such an approach, in this paper we elaborate a more challenging situation where the identity of the aggregators is unknown to the cluster members as well in Section 4.3. The clear advantage is that even if a node is compromised, its aggregator cannot be identified.

4. Advanced Protocol

The advanced private data aggregation protocol is designed to withstand the compromise of some nodes without revealing the identities of the aggregator. The protocol consists of four main parts. The first part is the initialization, which provides the required communication channel. The second part is needed for the data aggregator election. This subprotocol must ensure that the cluster does not remain without a cluster aggregator. This must be done without revealing the identity of the elected aggregator. The third part is needed for the data aggregation. This subprotocol must be able to forward the measured data to the aggregator without knowing its identifier. The last part must support the queries, where an operator queries some stored aggregated data.

Table 3: Summary of complexity of the advanced protocol. N is the number of nodes in the cluster.

	Election	Aggregation	Query
Message complexity	$O(N^2)$	$O(N)$	$O(N)$
Modular exponentiations	$4N$[1]	0	0
Hash computations	0	0	1

[1] 4 exponentiations for generating the two messages with knowledge proofs and 4N-4 exponentiations for checking the received knowledge proofs.

In the following, the description of each subprotocol follows the same pattern. First the goal and the requirements of the subprotocol are discussed, then the subprotocol itself is presented. After the presentation of the subprotocol, we analyze how it achieves its goal even in the presence of an attacker, and what data and services it provides for the next subprotocol.

At the end of this section, misbehavior is analyzed. We discuss, what an attacker can achieve, if its goal is not to identify the aggregators of the cluster, but to confuse the operation of the protocols.

In the following, it is assumed that every node knows which cluster it belongs to. The protocol descriptions are considering only one cluster, and separate instances of the protocol are run in different clusters independently.

The complexity of each subprotocol is summarized in Table 3.

4.1. Initialization. The initialization phase is responsible for providing the medium for authenticated broadcast communication. In the following, we shortly review the approaches of broadcast authentication in wireless sensor networks, and give some efficient methods for broadcast communication.

The initialization relies on some data stored on each node before deployment. Each node has some unique cryptographic credentials to enable authentication and is aware of the cluster identifier it belongs to. In the following, without further mentioning, we will assume, that each message contains the cluster identifier. Every message addressed to a cluster different from the one a node belongs to is discarded by the node. First, we briefly review the state of the art in broadcast authentication, then we propose a connected dominating set-based broadcast communication method, which fits well to the following aggregation and query phases.

4.1.1. Broadcast Authentication. Broadcast authentication enables a sender to broadcast some authenticated messages efficiently to a big number of potential receivers. In the literature, this problem is solved with either digital signatures or hash chains. In this section, we reviews some solutions from both approaches.

For the sake of completeness, Message Authentication Codes (MACs) must also be mentioned here [15]. MACs are based on symmetric cryptographic primitives, which enable very efficient computation. Unfortunately, the verifier of a MAC must also possess the same cryptographic credential the generator used for generating the MAC. It means that every node must know every credential in the network, to verify every message broadcast to the network. This full knowledge can be exploited by an attacker who compromises a node. The attacker can impersonate any other honest node, which means that if only one node is compromised, message authenticity can no longer be ensured.

One solution to the node compromise is the hop by hop authentication of the packets. In hop by hop authentication, every packets authentication information is regenerated by every forwarder. In this case, it is enough to only have a shared key with the direct neighbors of a node. In case of node compromise, only the node itself and the direct neighbors can be impersonated. Such a neighborhood authentication is provided by Zhu et al. in LEAP [9], where it is based on so-called cluster keys.

To make the authentication scheme robust against node compromise, one approach is the usage of asymmetric cryptography, namely, digital signatures.

Digital signatures are asymmetric cryptographic primitives, where only the owner of a private key can compute a digital signature over a message, but any other node can verify that signature. Computing a digital signature is a time-consuming task for a typical sensor node, but there exist some efficient elliptic curve-based approaches in the literature [16–19].

One of the first publicly available implementations was the TinyECC module written by Liu and Ning [16]. A more efficient implementation is the NanoECC module. Proposed by Szczechowiak et al. [17]. It is based on the MIRACL cryptographic library [20]. Up to now, to the best of our knowledge, the fastest implementations are the TinyPBC by Oliveira et al. [18], which is based on the RELIC toolkit [21], and the TinyPairing proposed by Xiong et al. in [19].

Another approach is proposed for broadcast authentication in wireless sensor networks by Perrig et al. in [22]. The μTESLA scheme is based on delayed release of hash chain values used in MAC computations. The scheme needs secure loose time synchronization between the nodes. The μTESLA scheme is efficient if it is used for authenticating many messages, but inefficient if the messages are sparse. Consequently, if only the rarely sent election messages must be authenticated, then the time synchronization itself can cause a heavier workload than simple digital signatures. If the aggregation messages must also be authenticated, then μTESLA can be an efficient solution. A DoS resistant version specially adapted for wireless sensor networks is proposed by Liu et al. in [23]. A faster but less secure modification is proposed by Huang et al. in [24].

In the following we will assume that an efficient broadcast authentication scheme is used without any indication.

4.1.2. Broadcast Communication. Broadcast communication is a method that enables sending information from one source to every other participant of the network. In wireless networks it can be implemented in many ways, like flooding the network or with a sequence of unicast messages.

A natural question would be, why broadcast communication is so important to the advanced protocol? The reason is that only broadcast communication can hide

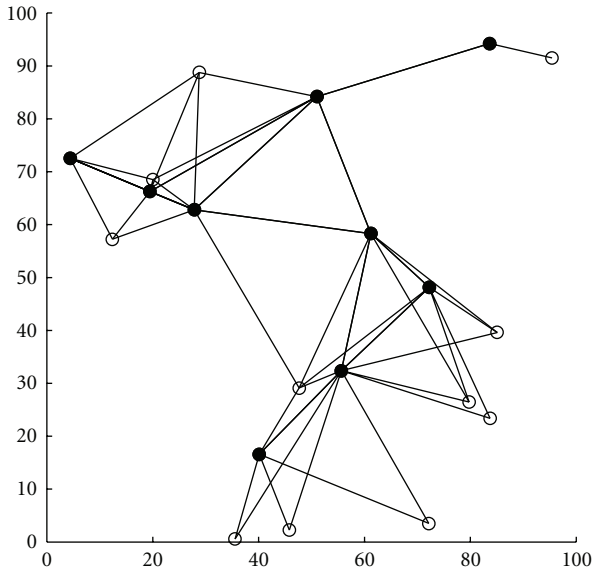

Figure 6: Connected dominating set. Solid dots represents the dominating set, and empty circles represent the remaining nodes. The connections between the non-CDS nodes of the network is not displayed on the figure.

the traffic patterns of the communication, thus not revealing any information about the aggregators.

An efficient way of implementing broadcast communication in wireless sensor networks is the usage of connected dominating set (CDS). The connected dominating set S of graph G is defined as a subset of G such that every vertex in $G - S$ is adjacent to at least one member of S, and S is connected. A graphical representation of a CDS can be found in Figure 6. The minimum connected dominating set (MDCS) is a connected dominating set with minimum cardinality. Finding an MDCS in a graph is an NP-Hard problem, however there are some efficient solutions which can find a close to minimal DCS in WSNs. For a thorough review of the state of the art of DCS in WSNs, the interested reader is referred to [25, 26].

In the following, we will assume that a connected dominating set is given in each cluster, and a minimum spanning tree is generated between the nodes in the CDS. Finding a minimum spanning tree in a connected graph is a well-known problem for decades. Efficient polynomial algorithms are suggested in [27, 28]. This kind of two layer communication architecture enables the efficient implementation of different kind of broadcast like communications, which are required for the following protocols. The spanning tree is used in the aggregation protocol in Section 4.3.

The simple all node broadcast communication can be implemented simply: if a node sends a packet to the broadcast address, then every node in the CDS forwards this message to the broadcast address. The CDS members are connected and every non-CDS member is connected to at least one CDS member by definition, so the message will be delivered to every recipient in the network. This approach is more efficient than simple flooding as only a subset of the nodes forwards the message, but the properties of the CDS ensures that every node in the cluster will eventually receive the broadcast information. Here, the notion of CDS parent (or simply parent) must be introduced. The CDS parent of node A is a node, which is in communication distance with A and is a member of the CDS.

The complexity of such a broadcast communication is $O(N)$, but actually it takes $|S|$ messages to broadcast some information, where $|S|$ is the number of nodes in the connected dominating set. If the CDS algorithm is accurate, than it can be very close to the minimum number of nodes required to broadcast communication.

In the following, we will use broadcast communication frequently to avoid that an attacker can gain some knowledge about the identity of the aggregators from the traffic patterns inside the network. Obviously we will not broadcast every message as it is in the network, because that would shortly lead to battery depletion and inoperability of the sensor network. Instead of automatically broadcasting every message, we will try to aggregate as much information as possible in each message to preserve energy. In the following sections, we will use the given CDS in different ways, and each particular usage will be described in the corresponding section.

The used communication patterns are closely related to and inspired by the Echo algorithm published by Chang in [29]. The Echo algorithm is a Wave algorithm [30], which enables the distributed computation of an idempotent operator in trees. It can be used in arbitrary connected graphs and generates a spanning tree as a side result.

4.2. Data Aggregator Election. The main goal of the aggregator node election protocol is to elect a node that can store the measurements of the whole cluster in a given epoch, but in such a way that the identity remains hidden. The election is successful if at least one node is elected. The protocol is unsuccessful if no node is elected, thus no node stores the data. In some cases, electing more than one node can be advantageous, because the redundant storage can withstand the failure of some nodes. In the following, we propose an election protocol, where the expected number of elected aggregators can be determined by the system operator, and the protocol ensures that at least one aggregator is always elected.

The election process relies on the initialization subprotocol discussed in Section 4.1. It requires an authenticated broadcast channel among the cluster members, which is exactly what the initialization part offers.

The election process consists of two main steps: (i) every node decides, whether it wants to be an aggregator, based on some random values. This step does not need any communication, the nodes compute the results locally. (ii) In the second step, an anonymous veto protocol is run, which reveals only the information that at least one node elected itself to be aggregator node. If no aggregator is elected, it will be clear for every participant, and every participant can run the election protocol again.

Step (i) can be implemented easily. Every node elects itself aggregator with a given probability p. The result of the election is kept secret, the participants only want to know that the number c of aggregators is not zero, without revealing the identity of the cluster aggregators. This is advantageous, because in case of node compromise, the attacker learns only whether the compromised node is an aggregator, but nothing about the identity or the number of the other aggregators. Let us denote the random variable representing the number of elected aggregators with C. It is easy to see that the distribution of C is binomial (N is the total number of nodes in one cluster):

$$\Pr(C = c) = \binom{N}{c} p^c (1-p)^{N-c}. \tag{6}$$

The expected number of aggregators after the first step is $c_E = Np$. So if on average $\hat{c}$ cluster aggregator is needed, then p should be $\hat{c}/N$ (this formula will be slightly modified after considering the results of the second step).

The probability that no cluster aggregator is elected is $(1-p)^N$.

To avoid the anarchical situation when no node is elected, the nodes must run step (ii) which proves that at least one node is elected as aggregator node, but the identity of the aggregator remains secret. This problem can be solved by an anonymous veto protocol. Such a protocol is suggested by Hao and Zieliński in [31].

Hao and Zieliński's approach has many advantageous properties compared to other solutions [32, 33], such as the property that it requires only 2 communication rounds.

The anonym veto protocol requires knowledge proofs. Informally, a knowledge proof allows a prover to convince a verifier that he knows a solution of a hard-to-solve problem without revealing any useful information about the knowledge. A detailed explanation of the problem can be found in [34].

A well known example of knowledge proof is given by Schnorr in [35]. The proposed method gives a noninteractive proof of knowledge of a logarithm without revealing the logarithm itself. The operation can be described briefly as follows. The proof of knowledge of the exponent of g_i^x consists of the pair $\{g^v, r = v - x_i h\}$, where $h = H(g, g^v, g_i^x, i)$ and H is a secure hash function. This proof of knowledge can be verified by anyone through checking whether g^v and $g^r g^{x_i h}$ are equal.

The operation of the anonym veto protocol consists of two consecutive rounds (G is a publicly agreed group with order q and generator g).

(1) First, every participant i selects a secret random value: $x_i \in \mathbb{Z}_q$. Then g_i^x is broadcast with a knowledge proof. The knowledge proof is needed to ensure that the participant knows x_i without revealing the value of x_i. Without knowledge proof, the node could choose g_i^x in a way to influence the result of the protocol (it is widely believed that for a given $g_i^x (\bmod p)$ it is hard to find $x_i (\bmod p)$, this problem is known as the discrete logarithm problem). Then every participant checks the knowledge proofs and computes a special product of the received values:

$$g^{y_i} = \frac{\prod_{j=1}^{i-1} g^{x_j}}{\prod_{j=i+1}^{N} g^{x_j}}. \tag{7}$$

(2) $g^{y_i c_i}$ is broadcast with a knowledge proof (the knowledge proof is needed to ensure that the node cannot influence the election maliciously afterwards). c_i is set to x_i for nonaggregators, while a random r_i value for aggregators.

The product $P = \prod_{i=1}^{N} g^{c_i y_i}$ equals 1 if and only if no cluster aggregator is elected (none vetoed the question: is the number of cluster aggregators elected zero?). If no aggregator is elected, then it will be clear for all participants, and the election can be done again. If P differs from 1, then some nodes are announced themselves to be cluster aggregators, and this is known by all the nodes.

If we consider the effect of the second step (new election is run if no aggregator is elected), the expected number of aggregators is slightly higher than in the case of binomial distributions. The expected number of aggregators are

$$c_E = \frac{Np}{1-(1-p)^N}. \tag{8}$$

The anonymity of the election subprotocol depends on the parts of the protocol. Obviously, the random number generation does not leak any information about the identity of the aggregator nodes, if the random number generator is secure. A cryptographically secure random number generator, called TinyRNG, is proposed in [36] for wireless sensor networks. Using a secure random number generator, it is unpredictable, who elects itself to be aggregator node.

The anonymity analysis of the anonym veto protocol can be found in [31]. The anonymity is based on the decisional Diffie-Hellman assumption, which is considered to be a hard problem.

The message complexity of the election is $O(N^2)$, which is acceptable as the election is run infrequently (N is the number of nodes in the cluster).

If this overhead with the 4 modular exponentiations (see Table 3 for the complexities and Table 1 for the estimated running times, note that RSA is based on modular exponentiation) is too big for the application, then it can use the basic protocol described in Section 3.1, where only symmetric key encryption is used.

In wireless sensor networks, the links in general are not reliable, packet losses occur in time to time. Reliability can be introduced by the link layer or by the application. As it is crucial to run the election protocol without any packet loss, it is required to use a reliable link layer protocol for this subprotocol. Such protocols are suggested in [37, 38] for wireless sensor networks.

As a summary, after the election subprotocol every node is equiprobably aggregator node. The election subprotocol ensures that at least one aggregator is elected and this node(s) is aware of its status. An outsider attacker does not know

the identity of the aggregators or even the actual number of the elected aggregator nodes. An attacker, who compromised one or more nodes, can decide whether the compromised nodes are aggregators, but cannot be certain about the other nodes.

4.3. Data Aggregation. The main goal of the WSN is to measure some data from the environment and store the data for later use. This section describes how the data is forwarded to the aggregator(s) without the explicit knowledge of the identifier(s) of the aggregator(s).

The data aggregation and storage procedure use the broadcast channel. If the covered area is so small or the radio range is so large that every node can reach each other directly, then the aggregation can be implemented simply. Every node broadcasts their measurement to the common channel, and the cluster aggregator(s) can aggregate and store the measurements. If the covered area is bigger (which is the more realistic case), a connected dominating set-based solution is proposed.

In each time slot, each ordinary node (not member of the CDS) sends its measurement to one neighboring CDS member (to the parent) by unicast communication. When the epoch is elapsed and all the measurements from the nodes are received, the CDS nodes aggregate the measurements and use a modification of the Echo algorithm on the given spanning tree to compute the gross aggregated measurement in the following way: each CDS member waits until all but one CDS neighbor sends its subaggregate to it, and after some random delay it sends the aggregate to the remaining neighbor. This means that the leaf nodes of the tree start the communication, and then the communication wave is propagated towards the root of the spanning tree. This behavior is the same as the second phase of the Echo algorithm. When one node receives the subaggregates from all of its neighbors, thus cannot send it to anyone, it can compute the gross aggregated value of the network. Then, this value is distributed between the cluster members by broadcasting it every CDS member.

This second phase is needed, so that every member of the cluster can be aware of the gross aggregated value, and the anonymous aggregators can store it, while the others can simply discard it. The stored data includes the time slot in which the aggregate was computed, and the environmental variables if more than one variable (e.g., temperature and humidity) is recorded besides the value itself.

The aggregation function can be any statistical function of the measured data. Some easily implementable and widely used functions are the minimum, maximum, sum, or average. In Figure 7, the aggregation protocol is visualized with five nodes and two aggregators using the average as an aggregation function.

The anonymity analysis of the aggregation subprotocol is quite simple. After the aggregation, every node possesses the same information as an external attacker can get. This information is the aggregated data itself, without knowing anything about the identity of the aggregators. If the operator wants to hide the aggregated data, it can use some techniques discussed in Section 5.

The message complexity of the aggregation is $O(N)$, where N is the number of nodes in the cluster. This is the best complexity achievable, because to store all the measurements by a single aggregator, all nodes must send the measurements towards the aggregator, which leads to $O(N)$ message complexity. In terms of latency, the advanced protocol doubles the time the aggregated measurement arrives to the aggregator compared to a naive system, where the identity of the aggregators is known to every participant. This latency is acceptable as in most WSN applications the time between the measurements is much longer than the time required to aggregate the data.

As mentioned in the election subprotocol, the protocol must be prepared to packet losses due to the nature of wireless sensor networks. In the aggregation subprotocol two kinds of packet loss can be envisioned: a packet can be lost before or after the final aggregate is computed. Both cases can be detected by timers and a resend request can be sent. If the resend is unsuccessful for some times, the aggregation must be run without those messages. If the lost message contains a measurement or subaggregate, then the final aggregate will be computed without that data leading to an inaccurate measurement. If the lost message contained the gross aggregate, then some nodes will not receive the gross aggregate. Here it is very useful that the network can have multiple aggregators, because if at least one aggregator received the data, the data can be queried by the operator.

4.4. Query. The ultimate goal of the sensor network is to make the measured data available to the operator upon request. While the aggregation subprotocol ensures that the measured data is stored by the aggregators, the goal of the query subprotocol is to provide the requested data to the operator and keep the aggregators' identity hidden at the same time.

One solution would be that the operator visits all the nodes and connects to them by wire. While this solution would leak no information about the identities of the aggregators to any eavesdropping attacker, the execution would be very time consuming and cumbersome. Moreover, the accessibility of some nodes may be difficult or dangerous (e.g., in a military scenario). Therefore, we propose a solution where it is sufficient for the operator to get in wireless communication range of any of the nodes. This node does not need to be an aggregator, as actually no one, not even the operator knows who the aggregator nodes are.

As a first step, the operator authenticates itself to the selected node O using the key k_O. After that, node O starts the query protocol by sending out a query, obtains the response to the query from the cluster, and makes the response available to the operator. In the following, we will assume that O is not a CDS node. (If it is indeed a CDS node, then the first and last transmission of the query protocol can be omitted.)

Node O broadcasts the query data Q with the help of the CDS nodes in the cluster. This is done by sending Q to the CDS parent, and then every CDS member rebroadcasts

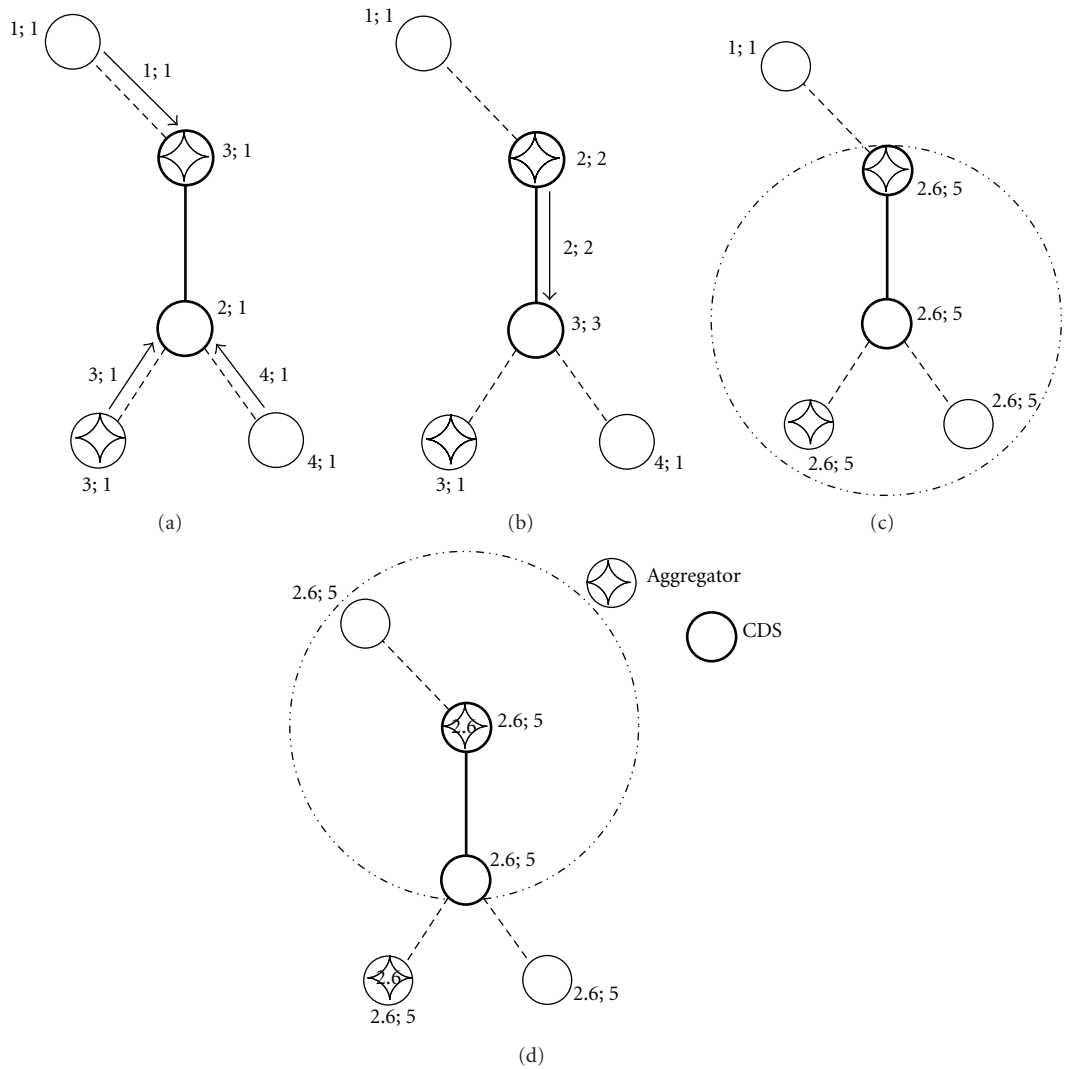

Figure 7: Aggregation example. The subfigures from left to right represents the consecutive steps of an average computation: (i) the measured data is ready to send. It is stored in a format of actual average; number of data. Non-CDS nodes sends the average to their parents. (ii) The CDS nodes start to send the aggregated value to its parents. (iii) A CDS node receives an aggregate from all of its neighbors, and starts to broadcast the final aggregated value. Nodes willing to store the value can do so. (iv) Other CDS nodes receiving the final value rebroadcasts it. Nodes willing to store the value can do so.

Q as it is received. The query Q describes what information the operator is interested in. It includes a variable name, a time interval, and a field for collecting the response to the query. It also includes a bit, called "aggregated", which will later be used in the detection of misbehaving nodes. For the details of misbehaving node detection, the reader is referred to Section 4.5; here we assume that the "aggregated" bit is always set meaning that aggregation is enabled.

The idea of the query protocol is that each node i in the cluster contributes to the response by a number R_i, which is computed as follows:

$$R_i = \begin{cases} h(Q \mid k_i), & \text{for nonaggregators,} \\ h(Q \mid k_i) + M, & \text{for aggregators,} \end{cases} \tag{9}$$

where M is the stored measurement (available only if the node is an aggregator), h is a cryptographic hash function, and k_i is the key shared by node i and the operator. Thus, nonaggregators contribute with a pseudorandom number $h(Q \mid k_i)$ computed from the query and the key k_i, which can later be also computed by the operator, while aggregator nodes contribute with the sum of a pseudorandom number and the requested measurement data. The sum is normal fix point addition, which can overflow if the hash is a large value.

The goal is that the querying node O receives back the sum of all these R_i values. For this reason, when the query Q is received by a non-CDS node from its CDS parent, it computes its R_i value and sends it back to the CDS parent in the response field of the query token. When a CDS parent receives back the query tokens with the updated response field from its children, it computes the sum of the received

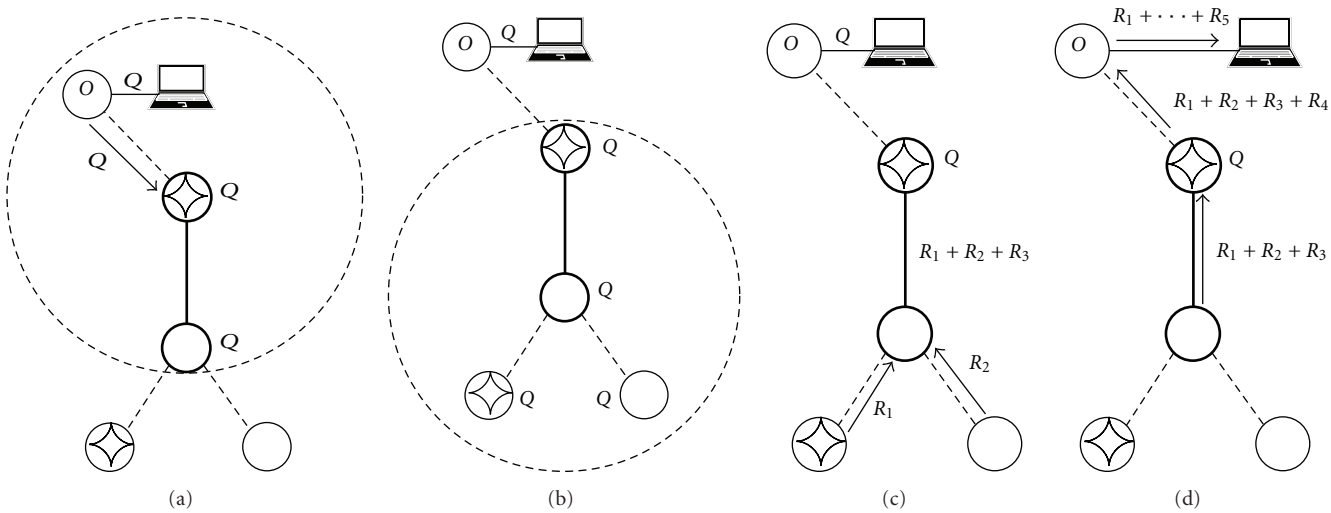

Figure 8: Query example. The subfigures from left to right represents the consecutive steps of a query: (i) the operator sends the Q query to node O. This node forwards it to its CDS parent. The CDS parent broadcasts the query. (ii) The CDS nodes broadcasts the query, so every node in the network is aware of Q. (iii) Every non-CDS node (except O) sends it response to its parent. (iv) The sum of the responses is propagated back to the parent of O (including the list of responding nodes, not on the figure), who forwards it to the operator through O.

R_i values and its own, and after inserting the identifiers of the nodes sends the result back to its parent. This is repeated until the query token reaches back to the CDS parent of node O, which can forward the response $R = \sum R_i$ and the list of responding nodes to node O, where the sum is computed by normal fix point addition. This operation is illustrated in Figure 8.

When receiving R from O, the operator can calculate the stored data as follows. First of all, the operator can regenerate each hash value $h(Q \mid k_i)$, because it stores (or can compute from a master key on-the-fly) each key k_i, and it knows the original query data Q. The operator can subtract the hash values from R (note that the responding nodes list is present in the response), and it gets a result $R' = cM$, where c is the actual number of aggregators in the cluster. (Note that each aggregator contributed the measurement M to the response, that is why at the end, the response will be c times M, where c is the number of aggregators.) Unfortunately, this number c is unknown to the operator, as it is unknown to everybody else. Nevertheless, if M is restricted to lie in an interval $[A, B]$ such that the intervals $[iA, iB]$ for $i = 1, 2, \ldots, N$ are nonoverlapping, then cM can fall only into interval $[cA, cB]$, and hence, c can be uniquely determined by the operator by checking which interval R' belongs to. Then, dividing R' with c gives the requested data M.

More specifically, and for practical reasons, the following three criteria need to be satisfied by the interval $[A, B]$ for our query scheme to work: (i) as we have seen before, for unique decoding of cM, the intervals $[iA, iB]$ for $i = 1, 2, \ldots, N$ must be nonoverlapping, (ii) in order to fit in the messages and to avoid integer overflow (In case of overflow, the result is not unique.), the highest possible value for cM, that is, NB must be representable with a prespecified number L, and (iii) it must be possible to map a prespecified number D of different values into $[A, B]$.

The first criterion (i) is met, if the lower end of each interval is larger than the higher end of the preceding interval:

$$0 < iA - (i-1)B = i(A - B) + B, \qquad i = 1, \ldots, N. \tag{10}$$

Note that if the above inequality holds for $i = N$, then it holds for every i, because $A - B$ is a negative constant and B is a positive constant. So it is enough to consider only the case of $i = N$:

$$\begin{aligned} 0 &< N(A - B) + B, \\ B &< \frac{N}{N-1} A. \end{aligned} \tag{11}$$

The second criterion (ii) means that

$$\begin{aligned} BN &< L, \\ B &< \frac{L}{N} \end{aligned} \tag{12}$$

while the third criterion (iii) can be formalized as

$$\begin{aligned} D &< B - A, \\ B &> A + D. \end{aligned} \tag{13}$$

Figure 9 shows an example for a graphical representation of the three criteria, where the crossed area represents the admissible (A, B) pairs. It can also be easily seen in this figure that a solution exists only if the B coordinate of the intersection of inequalities (11) and (13) meets criterion (12), or in other words

$$NM < \frac{L}{N}. \tag{14}$$

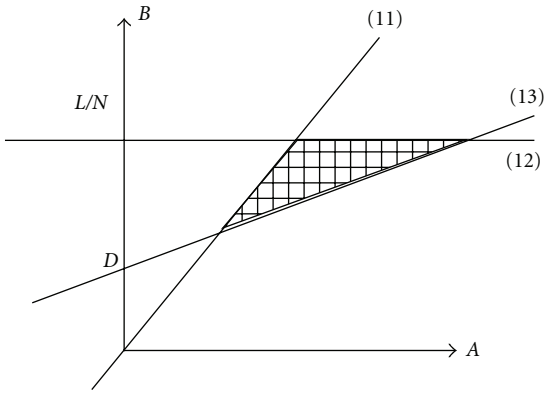

Figure 9: Graphical representation of the suitable intervals.

As a numerical example, let us assume that we want to measure at least 100 different values ($D = 99$); the microcontroller is a 16 bit controller ($L = 2^{16}$), and we have at most 20 nodes in each cluster ($N = 20$). Then a suitable interval that satisfies all three criteria would be $[A, B] = [2000–2100]$. Checking that this interval indeed meets the requirements is left for the interested reader. Finally, note that any real measurement interval can be easily mapped to this interval $[A, B]$ by simple scaling and shifting operations, and our solution requires that such a mapping is performed on the real values before the execution of the query protocol.

Our proposed protocol has many advantageous properties. First, the network can respond to a query if at least one aggregator can successfully participate in the subprotocol. Second, the operator does not need to know the identity of the aggregators, thus even the operator cannot leak that information accidentally (although, after receiving the response, the operator learns the actual number of the aggregator nodes). Third, the protocol does not leak any information about the identity of the aggregators: an attacker can eavesdrop the query information Q, and the R_i pseudo random numbers, but cannot deduce from them the identity of the aggregators. Finally, the message complexity of the query is $O(N)$, where N is the number of nodes in the cluster. This is the best complexity achievable, when the originator of the query does not know the identity of the aggregator(s). The latency of the query protocol depends on the longest path of the network rooted at node O.

As mentioned in the previous subprotocols, the protocol must be prepared to packet losses due to the nature of wireless sensor networks. Due to the packet losses, the final sum R is the sum of the responding nodes which is a subset of all nodes. That is why the identifiers must be included in the responses. The operator can calculate cM independently from the actual subset of responders. If at least one response from an aggregator gets to the operator, it can calculate M in the previously described way. If $cM = 0$, then it is clear for the operator that every aggregators' response is lost.

4.5. Misbehaving Nodes. In this section, we look beyond our initial goal. We briefly analyze what happens if a compromised node deviates from the protocol to achieve some goals other than just learning the identity of the aggregators.

In the election process, a compromised node may elect itself to be aggregator in every election. This can be a problem if this node is the only elected aggregator, because a compromised node may not store the aggregated values. Unfortunately this situation cannot be avoided in any election protocol, because an aggregator can be compromised after the election, and the attacker can erase the memory of that node. Actually our protocol is partially resistant to this attack, because more than one aggregator may be elected with some probability, and the attacker cannot be sure if the compromised node is the single aggregator node in the cluster.

During the aggregation, a misbehaving node can modify its readings, or modify the values it aggregates. The modification of others' values can be prevented by some broadcast authentication schemes discussed in Section 4.1.1. The problem of reporting false values can be handled by statistical approaches discussed in [39–41].

The most interesting subprotocol from the perspective of misbehaving nodes is the query protocol. In this protocol, a compromised node can easily modify the result of the query in the following way. A compromised node can add an arbitrary number X to the hash in (9) instead of using 0 or M. It is easy to see, that if X is selected from the interval $[A, B]$, then after subtracting the hashes, the resulting sum R' will be an integer in the interval $[(c + 1)A, (c + 1)B]$ (c is the actual number of aggregators, $c + 1$ nodes act like aggregators, the c aggregator, and the compromised node). A compromised node can further increase its influence by choosing X from the interval $[iA, iB]$. This means that the resulting sum R' will be in the interval $[(c + i)A, (ci)B]$. If X is not selected from interval $[jA, jB]$, $j = 1, \ldots, N$, then the result can be outside of the decodable intervals. This can be immediately detected by the operator (see Figure 10).

If the result is in a legitimate interval ($\exists j, R' \in [jA, jB]$), then the operator can further check the consistency by calculating $R' \bmod j$. If the result is zero, then it is possible, that no misbehaving node is present in the network. If the result is nonzero, the operator can be sure, that apart from the zeros and Ms, some node sent a different value, thus a misbehaving node is present in the network. It is hard for the attacker to guess j, because it neither knows the actual number of aggregators, nor can calculate R' from R by subtracting the unknown hashes.

If the modulus is zero, but the operator is still suspicious about the result, it can further test the cluster for misbehaving nodes with the help of the aggregated bit in the queries. This further testing can be done regularly, randomly, or on receiving suspicious results. If the aggregated bit is cleared in a query Q, then the CDS nodes doesnot sum the incoming replies, but forward them towards the agent O node as they are received. So if the operator wants to check if a misbehaving node is present in the network it can run a query Q with aggregated bit set, and then run the same query with cleared aggregated bit. If the two results are different, then the operator can be sure that a node wants to hide its malicious activity from the operator. If the two

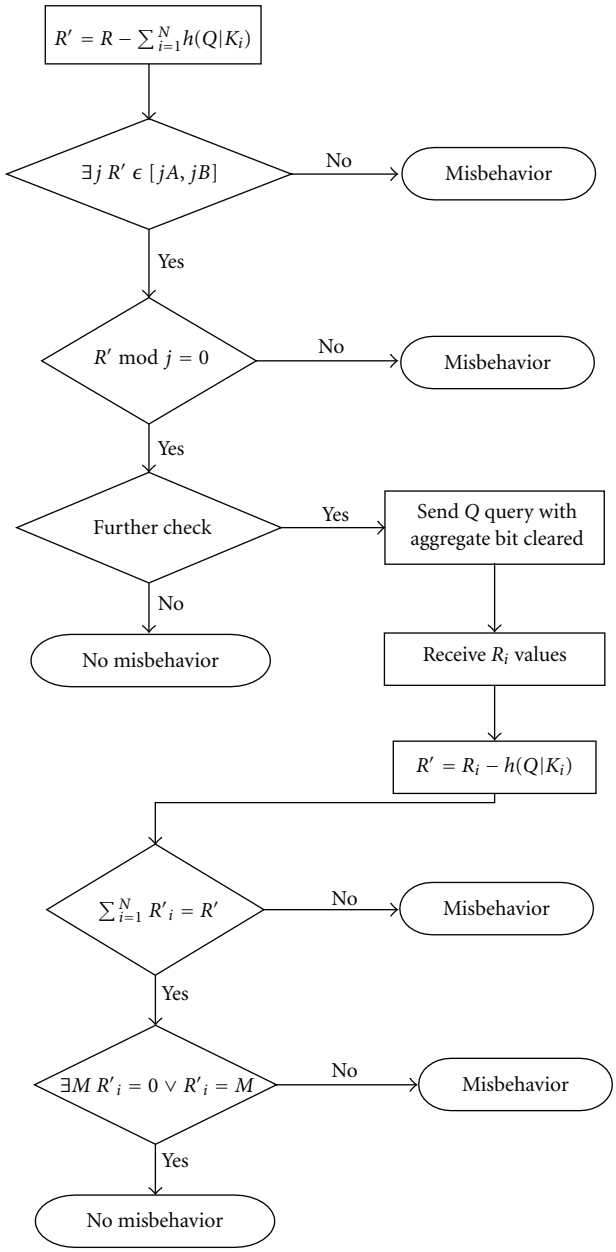

Figure 10: Misbehavior detection algorithm for the query protocol.

sums are equal, then the operator can further check the results from the second round. If the values are all equal after subtracting the hashes (not considering the zero values), then no misbehavior is detected, otherwise some node(s) misbehave in the cluster.

Note here that this algorithm does not find every misbehavior, but the misbehaviors not detected by this algorithm does not influence the operator. For example, two nodes can misbehave such that the first adds S to its hash and the second adds $-S$. It is clear that this misbehavior does not affect the result computed by the operator, because $S - S = 0$. Other misbehavior is not detected by the algorithm if a compromised non aggregator node sends M instead of 0. This is not detected by the algorithm, but doesnot modify the result the operator computes. The operation of misbehavior detection algorithm is depicted on Figure 10. This algorithm only detects if some misbehavior occurrs in the cluster, but does not necessarily find the misbehaving node. We left the elaboration of this problem for future work.

5. Related Work

A survey on privacy protection techniques for WSNs is provided in [42], where they are classified into two main groups: data-oriented and context-oriented protection. In this section, we briefly review these techniques, with an emphasis on those solutions that are closely related to our work.

In data-oriented protection, the confidentiality of the measured data must be preserved. It is also a research direction how the operator can verify if the received data is correct. The main focus is on the confidentiality in [43], while the verification of the received data is also ensured in [44].

According to [42] context-oriented protection covers the location privacy of the source and the base station. The source location privacy is mainly a problem in event-driven networks, where the existence and location of the event is the information, which must be hidden. The location privacy of the base station is discussed in [45]. The main difference between hiding the base station and the in-network aggregators is that a WSN regularly contains only one base station which is a predefined node, while at the same time there are more in network aggregators used in one network, and the nodes used as aggregators are periodically changed.

The problem of private cluster aggregator election in wireless sensor networks is strongly related to anonym routing in WSNs. The main difference between anonym routing and anonymous aggregation is that anonym routing supports any traffic pattern and generally handles external attackers, while anonymous aggregation supports aggregation-specific traffic patterns and can handle compromised nodes as well. In [12] an efficient anonymous on demand routing scheme called ARM is proposed for mobile ad hoc networks. For the same problem another solution is given in [13] (MASK), where a detailed simulation is also presented for the proposed protocol. A more efficient solution is given in [14], which uses low cryptographic overhead, and addresses some drawbacks of the two papers above. In [46] a privacy preserving communication system (PPCS) is proposed. PPCS provides a comprehensive solution to anonymize communication endpoints, keep the location and identifier of a node unlinkable, and mask the existence of communication flows.

The basis of this paper are two conference papers [1, 2]. Reference [1] describes the basic protocol discussed in Section 3, while [2] describes the advanced protocol discussed in Section 3. The main difference between this paper and [2], is that here a connected dominating set-based solution is proposed, while the previous paper assumed the existence of a Hamilton cycle inside the cluster. The management of such a cycle can be problematic in WSNs, while the CDS-based broadcast communication can be efficiently implemented in such a network [25]. Another contribution of this paper is the misbehavior detection algorithm, which solves a problem not discussed previously.

6. Conclusion

In wireless sensor networks, in-network data aggregation is often used to ensure scalability and energy efficient operation. However, as we saw, this also introduces some security issues: the designated aggregator nodes that collect and store aggregated sensor readings and communicate with the base station are attractive targets of physical node destruction and jamming attacks. In order to mitigate this problem, in this paper, we proposed two private aggregator node election protocols for wireless sensor networks that hide the elected aggregator nodes from the attacker, who, therefore, cannot locate and disable them. Our basic protocol provides fewer guarantees than our advanced protocol, but it may be sufficient in cases where the risk of physically compromising nodes is low. Our advanced protocol hides the identity of the elected aggregator nodes even from insider attackers, thus it handles node compromise attacks too.

We also proposed a private data aggregation protocol and a corresponding private query protocol for the advanced version, which allow the aggregator nodes to collect sensor readings and respond to queries of the operator, respectively, without revealing any useful information about their identity. Our aggregation and query protocols are resistant to both external eavesdroppers and compromised nodes participating in the protocol. The communication in the advanced protocol is based on the concept of connected dominating set, which suits well to wireless sensor networks.

In this paper we went beyond the goal of only hiding the identity of the aggregator nodes. We also analyzed what happens if a malicious node wants to exploit the anonymity offered by the system and tries to mislead the operator by injecting false reports. We proposed an algorithm that can detect if any of the nodes misbehaves in the query phase. We only detect the fact of misbehavior and leave the identification of the misbehaving node itself for future work.

In general, our protocols increase the dependability of sensor networks, and therefore, they can be applied in mission critical sensor network applications, including high-confidence cyber-physical systems where sensors and actuators monitor and control the operation of some critical physical infrastructure.

Disclosure

The information in this document is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

Acknowledgments

The work described in this paper is based on results of the WSAN4CIP Project (http://www.wsan4cip.eu), which receives research funding from the European Community's 7th Framework Programme. Apart from this, the European Commission has no responsibility for the content of this paper. The second author has also been supported by the Hungarian Academy of Sciences through the Bolyai János Research Fellowship. The authors are thankful to the anonymous reviewers for their useful comments and suggestions that helped to improve the quality of this paper.

References

[1] L. Buttyán and T. Holczer, "Private cluster head election in wireless sensor networks," in *Proceedings of the 6th IEEE*

International Conference on Mobile Adhoc and Sensor Systems (MASS '09), pp. 1048–1053, 2009.

[2] L. Buttyán and T. Holczer, "Perfectly anonymous data aggregation in wireless sensor networks," in *Proceedings of the 7th IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS '10)*, pp. 513–518, 2010.

[3] Y. R. Faizulkhakov, "Time synchronization methods for wireless sensor networks: a survey," *Programming and Computer Software*, vol. 33, no. 4, pp. 214–226, 2007.

[4] L. Buttyán and P. Schaffer, "Position-based aggregator node election in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 2010, Article ID 679205, 15 pages, 2010.

[5] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proceedings the 33rd Annual Hawaii International Conference on System Siences (HICSS '00)*, vol. 2, pp. 3005–3014, January 2000.

[6] R. Anderson and M. Kuhn, "Tamper resistance: a cautionary note," in *Proceedings of the 2nd USENIX Workshop on Electronic Commerce*, vol. 2, p. 1, USENIX Association, 1996.

[7] J. Lopez and J. Zhou, *Wireless Sensor Network Security*, Cryptology and Information Security Series, IOS Press, 2008.

[8] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 197–213, IEEE Computer Society, May 2003.

[9] S. Zhu, S. Setia, and S. Jajodia, "LEAP: efficient security mechanisms for large-scale distributed sensor networks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*, pp. 62–72, ACM Press, October 2003.

[10] P. Ganesan, R. Venugopalan, P. Peddabachagari, A. Dean, F. Mueller, and M. Sichitiu, "Analyzing and modeling encryption overhead for sensor network nodes," in *Proceedings of the 2nd ACM International Workshop on Wireless Sensor Networks and Applications (WSNA '03)*, pp. 151–159, September 2003.

[11] K. Piotrowski, P. Langendoerfer, and S. Peter, "How public key cryptography influences wireless sensor node lifetime," in *Proceedings of the 4th ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '06)*, pp. 169–176, October 2006.

[12] S. Seys and B. Preneel, "ARM: anonymous routing protocol for mobile ad hoc networks," in *Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA '06)*, pp. 133–137, April 2006.

[13] Y. Zhang, W. Liu, W. Lou, and Y. Fang, "MASK: anonymous on-demand routing in mobile ad hoc networks," *IEEE Transactions on Wireless Communications*, vol. 5, no. 9, pp. 2376–2385, 2006.

[14] T. Rajendran and K. V. Sreenaath, "Secure anonymous routing in ad hoc networks," in *Proceedings of the 1st ACM Bangalore Annual Conference (COMPUTE '08)*, ACM Press, New York, NY, USA, January 2008.

[15] B. Preneel and P. C. Van Oorschot, "On the security of iterated message authentication codes," *IEEE Transactions on Information Theory*, vol. 45, no. 1, pp. 188–199, 1999.

[16] A. Liu and P. Ning, "TinyECC: a configurable library for elliptic curve cryptography in wireless sensor networks," in *Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN '08)*, pp. 245–256, April 2008.

[17] P. Szczechowiak, L. B. Oliveira, M. Scott, M. Collier, and R. Dahab, "NanoECC: testing the limits of elliptic curve cryptography in sensor networks," in *Proceedings of the 5th European Conference on Wireless Sensor Networks (EWSN '08)*, vol. 4913 of *Lecture Notes in Computer Science*, pp. 305–320, 2008.

[18] L. B. Oliveira, M. Scott, J. López, and R. Dahab, "TinyPBC: pairings for authenticated identity-based non-interactive key distribution in sensor networks," in *Proceedings of the 5th International Conference on Networked Sensing Systems (INSS '08)*, pp. 173–180, IEEE, Kanazawa, Japan, June 2008.

[19] X. Xiong, D. S. Wong, and X. Deng, "TinyPairing: a fast and lightweight pairing-based cryptographic library for wireless sensor networks," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC '10)*, April 2010.

[20] http://www.shamus.ie/.

[21] http://code.google.com/p/relic-toolkit/.

[22] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, "The TESLA broadcast authentication protocol," *RSA CryptoBytes*, vol. 5, 2002.

[23] D. Liu, P. Ning, S. Zhu, and S. Jajodia, "Practical broadcast authentication in sensor networks," in *Proceedings of the 2nd Annual International Conference on Mobile and Ubiquitous Systems -Networking and Services (MobiQuitous '05)*, pp. 118–129, July 2005.

[24] Y. Huang, W. Ret, and K. Nahrstedt, "ChainFarm: a novel authentication protocol for high-rate any source probabilistic broadcast," in *Proceedings of the 6th IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS '09)*, pp. 264–273, October 2009.

[25] J. Blum, M. Ding, A. Thaeler, and X. Cheng, "Connected dominating set in sensor networksand manets," in *Handbook of Combinatorial Optimization*, D.-Z. Du and P. Pardalos, Eds., pp. 329–369, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2004.

[26] P. Jacquet, "Performance of connected dominating set in olsr protocol," Tech. Rep. RR-5098, INRIA, 2004.

[27] J. Kruskal and B. Joseph, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical Society*, vol. 7, no. 1, pp. 48–50, 1956.

[28] R. Prim, "Shortest connection networks and some generalizations," *Bell System Technical Journal*, vol. 36, no. 6, pp. 1389–1401, 1957.

[29] E. J. H. Chang, "Echo algorithms: depth parallel operations on general graphs," *IEEE Transactions on Software Engineering*, vol. SE-8, no. 4, pp. 391–401, 1982.

[30] G. Tel, *Introduction to Distributed Algorithms*, Cambridge University Press, Cambridge, UK, 2nd edition, 2000.

[31] F. Hao and P. Zieliński, "A 2-round anonymous veto protocol," in *Proceedings of the 4th International Workshop on Security Protocols: Putting the Human Back in the Protocol*, vol. 5087 of *Lecture Notes in Computer Science*, pp. 202–211, Cambridge, UK, 2009.

[32] F. Brandt, "Efficient cryptographic protocol design based on distributed El gamal encryption," in *Proceedings of the 8th International Conference on Information Security and Cryptology (ICISC '05)*, vol. 3935 of *Lecture Notes in Computer Science*, pp. 32–47, 2006.

[33] D. Chaum, "The dining cryptographers problem: unconditional sender and recipient untraceability," *Journal of Cryptology*, vol. 1, no. 1, pp. 65–75, 1988.

[34] J. Camenisch and M. Stadler, "Proof systems for general statements about discrete logarithms," Tech. Rep., Department of Computer Science, ETH Zürich, Zürich, Switzerland, 1997.

[35] C. P. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, vol. 4, no. 3, pp. 161–174, 1991.

[36] A. Francillon and C. Castelluecia, "TinyRNG: a cryptographic random number generator for wireless sensors network nodes," in *Proceedings of the5th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt '07)*, cyp, April 2007.

[37] A. Iqbal and S. A. Khayam, "An energy-efficient link layer protocol for reliable transmission over wireless networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2009, Article ID 791201, 10 pages, 2009.

[38] C.-Y. Wan, A. T. Campbell, and L. Krishnamurthy, "PSFQ: a reliable transport protocol for wireless sensor networks," in *Proceedings of the ACM International Workshop on Wireless Sensor Networks and Applications*, pp. 1–11, ACM Press, 2002.

[39] L. Buttyán, P. Schaffer, and I. Vajda, "RANBAR: RANSAC-based resilient aggregation in sensor networks," in *Proceedings of the 4th ACM Workshop on Security of ad hoc and Sensor Networks (SASN '06)*, pp. 83–90, ACM Press, Alexandria, Va, USA, 2006.

[40] D. Wagner, "Resilient aggregation in sensor networks," in *Proceedings of the ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '04)*, pp. 78–87, October 2004.

[41] L. Buttyán, P. Schaffer, and I. Vajda, "CORA: correlation-based resilient aggregation in sensor networks," *Ad Hoc Networks*, vol. 7, no. 6, pp. 1035–1050, 2009.

[42] N. Li, N. Zhang, S. K. Das, and B. Thuraisingham, "Privacy preservation in wireless sensor networks: a state-of-the-art survey," *Ad Hoc Networks*, vol. 7, no. 8, pp. 1501–1514, 2009.

[43] W. He, X. Liu, H. Nguyen, K. Nahrstedt, and T. Abdelzaher, "PDA: privacy-preserving data aggregation in wireless sensor networks," in *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM '07)*, pp. 2045–2053, May 2007.

[44] B. Sheng and Q. Li, "Verifiable privacy-preserving range query in two-tiered sensor networks," in *Proceedings of the 27th IEEE Communications Society Conference on Computer Communications (INFOCOM '08)*, pp. 457–465, April 2008.

[45] J. Deng, R. Han, and S. Mishra, "Decorrelating wireless sensor network traffic to inhibit traffic analysis attacks," *Pervasive and Mobile Computing*, vol. 2, no. 2, pp. 159–186, 2006.

[46] H. Choi, P. McDaniel, and T. F. La Porta, "Privacy preserving communication in MANETs," in *Proceedings of the 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON '07)*, pp. 233–242, June 2007.