

Non-Manipulable Aggregator Node Election Protocols for Wireless Sensor Networks

Michael Sirivianos
University of California, Irvine

msirivia@uci.edu

Dirk Westhoff Frederik Armknecht Joao Girao
NEC Europe Ltd

{dirk.westhoff, frederik.armknecht, joao.girao}@netlab.nec.de

Abstract - *Aggregator nodes commonly have the ability to read, corrupt or disrupt the flow of the information produced by a Wireless Sensor Network. Despite this fact, existing aggregator node election schemes do not address an adversary that strives to influence the election process towards candidates that it controls. We discuss the requirements that need to be fulfilled by a non-manipulable aggregator node election protocol. We conclude that these requirements can be satisfied by a distributed random number generator function in which no node is able to determine the output of the function. We provide three protocols that instantiate such function and we evaluate them.*

I. Introduction

Cluster head node election protocols such as LEACH [14], HEED [29] and VCA [21], aim at flatly balancing and reducing energy consumption in order to extend a Wireless Sensor Network's (WSN) lifetime.

In this work, we focus on one type of cluster head node functionality: aggregation. Owing to an aggregator node's enhanced ability to eavesdrop on, pollute or disrupt the flow of the information produced by the sensor network, an adversary is highly incited to be in control of aggregator nodes. However, none of the aggregator node election protocols for WSNs proposed so far, accounts for communication partners that are motivated to increase the likeness of being elected in subsequent epochs. Although schemes aiming at preserving confidentiality [2, 6, 25] and integrity [9, 20] of aggregated data have already been proposed, these are suitable only for certain types of aggregation functions.

In existing aggregator election protocols, nodes rely on their local status or on reported status of peer nodes for electing their aggregator. For example, in LEACH and HEED, a node can independently decide to become aggregator based on its energy level. However, such approaches have substantial limitations with respect to security. A fundamental observation is that any election mechanism that is based on a concrete *election metric*, e.g. the residual energy level, can in principle be manipulated. The adversary is in position to feed values into the decision process that influence the aggregator election for the upcoming epoch in a way that is profitable for him. The above

observation motivates the statement that probably the most secure aggregator node election process is the one in which a node's input is as good as the other and ideally the election criterion is fully random. Hence, we design and compare secure node election protocols which: **i)** randomly choose the aggregator node in a decentralized way; and **ii)** use lightweight cryptographic primitives to ensure that no party can manipulate the outcome of the election process at honest nodes. In this extended abstract, we provide two protocols from such class of protocols.

II. Aggregator Node Election

We define an Aggregator Node Election protocol for WSNs as follows:

Definition 2.1: An *Aggregator Node Election* protocol is a distributed algorithm running in microsensor nodes that belong to a predefined and finite set S . The purpose of this algorithm is nodes in set S to reach consensus in electing a particular node $A \in S$ for a particular epoch t .

An aggregator node election protocol's ultimate goal is to prolong the lifetime of the sensor network. We note that a WSN may still work properly even if some nodes become exhausted.

Definition 2.2: The *network lifetime* of a WSN is the latest epoch in which the set of non-exhausted nodes can effectively communicate the information produced by the sensor network to its consumer.

Hence, an aggregator node election protocol in microsensor networks strives to flatly balance the energy consumption of the network, while maintaining connectivity between the clusters and the consumer of the WSN's information. It supports this by carefully assigning to nodes the energy consuming aggregator task. One can derive two classes of existing aggregator node election protocols: a) protocols in which the election criterion is the remaining energy of the nodes. Nodes with higher remaining energy are more likely to be elected as the aggregator nodes [5, 14, 21, 29]; b) protocols in which nodes with high connectivity (i.e. many one-hop neighbors) are favored to be the aggregator node [3, 8, 16]. Regardless of whether an aggregator node election protocol belongs to the first or to the second category we observe:

- In the *non-critical* phase of the WSN's lifetime no nodes or a few nodes are exhausted or are close to be exhausted. Due to the WSN's density, connectivity is almost always ensured. The only critical task of an aggregator node election protocol should be to ensure that energy levels are *flatly* distributed over all nodes. This can be done by a decision criterion that considers the nodes' residual energy or even by a uniformly *random* election among the non-exhausted nodes. In this phase, *connectivity-based* approaches may not be necessary and result in uneven energy distribution.
- In the *critical* phase of the WSN's lifetime, a significant fraction of nodes is already exhausted and another significant fraction a is close to exhaustion; The *residual-energy-based* election criterion ensures that more nodes remain alive longer, as it refrains from assigning the aggregator task to a node that is close to energy depletion. The *connectivity-based* election criterion provides better connectivity guarantees because it elects nodes that are more likely to have non-exhausted neighbors. However, similar to the non-critical phase, it results in highly connected nodes becoming exhausted earlier. It may even be the case that the WSN has become so sparse that its physical connectivity cannot be achieved and the system's lifetime ends. With *randomized* election, for large a , the probability of electing nodes close to exhaustion becomes significant. This results in reduction of the system's lifetime and builds up situations where an election process restart is mandatory as the aggregator dies out. Consequently, the energy levels of the system are further reduced. With both *residual-energy-based* and *randomized* election, connectivity is not ensured, since a large portion of the network is exhausted.

In summary, depending on the ratio of the durations of the non-critical and critical phase, residual-energy-based and connectivity-based approaches may be a conceptual overkill with the disadvantage of being susceptible to adversarial manipulation. In the full version of this paper we defend these statements presenting our simulation results. Our observations indicate that the functional disadvantages of *randomized* aggregator node election mechanisms are not as compelling as one would expect. This motivates us to propose a family of (pseudo)random election protocols to increase the WSN's security, without adversely affecting its lifetime.

III. Assumptions and Objective

Network Model - We assume a CSMA/CA MAC scheme (such as IEEE 802.15.4) for broadcast and unicast communication. We further assume a fully decentralized network of equal functionality and capability sensor nodes. All nodes in the network are stationary. Each node can have the role of a sensing node, an aggregator node or a forwarding node. The sensor nodes are scattered over a large area so that they form small sets of nodes in close proximity from each other. We call these sets *sectors*. Nodes in the same sector are pre-configured with the same sector ID and

are said to belong to the same set S . For example, sensor nodes released from the same parachute would belong to the same sector. Election protocol messages are exchanged only among nodes in the same sector. These messages are either sent via unicast among nodes in the same sector or disseminated to all nodes in a sector using a simple *sector-aware* controlled flooding scheme. With this flooding scheme, nodes re-broadcast first-seen messages only if the source of the message belongs to the same sector as they do.

We denote the i -th sensor node in a sector S by s_i , and the aggregator node during the t -th epoch as A_t . For each sensor s_i , we denote by $N_i \subseteq S$ the set of nodes from which it has received valid election contributions during an election round. A contribution refers to a node's input to the election process. An election contribution of a node can be sent in one or two messages, or it can be aggregated with other node contributions as they are propagated in the sector.

Threat Model - The adversary is capable of eavesdropping on sensor nodes' transmissions, inject bits in the channel and spoof the source of its message. It is able to deploy adversarial nodes either by inserting its own nodes or compromising existing legitimate nodes. Adversarial nodes may collude to attack the system and may have very high bandwidth and residual energy. We do not assume any sensor nodes to be tamper resistant.

The most compelling incentive for cheating during node election is to influence honest nodes towards electing an adversarial node. Under our threat model an attacker can control any strict subset $F \subset S$ of nodes in sector S . For each node s_i , we denote by F_i , where $F_i \subseteq F$ and $F_i \subset N_i$, the set of nodes that are under adversarial control and participate in the election process at node s_i .

Remark: We call an adversary successful in *manipulating* an aggregator election protocol (definition 2.1), if he is able to manipulate the election at any node s_i , such that the probability that s_i elects an adversarial node is higher than $P_{suc} = |F_i|/|N_i|$.

Objective - At this point, we revisit the definition of an Aggregator Node Election protocol to address the above network and threat model.

Definition 3.1: A *Secure Aggregator Node Election (SANE)* protocol is a distributed algorithm running in microsensor nodes that belong to a predefined and finite set S . The set S may include adversarial nodes. Its purpose is to ensure that all non-exhausted nodes in set S , which have received the same set of valid election contributions for epoch t , reach consensus in electing a particular node $A_t \in S$. Their decisions should *not be manipulated*. towards the election of adversarial nodes.

It may appear that a SANE protocol solves a flavor of the Byzantine Agreement (BA) problem [17]. BA is a decision process in a distributed system aiming at a consensus in the presence of a subset of misbehaving parties and faulty communication channels. In a successful BA process, all honest nodes must agree on one reasonable

value derived from the values proposed by the participating parties. It requires that all parties receive the same set of proposed values. Flavors of this problem assume certain properties of the messages carrying the proposed values (e.g. whether their source can be authenticated) and certain characteristics of the communication channel. In contrast to the BA problem, for SANE we relax the requirement of *agreement*. In particular, we require that all honest nodes must agree on a common value only if they receive the same set of contributed values. We relax the agreement requirement in the context of WSNs for the following three reasons. First, failure to reach agreement about the next aggregator does not affect the correct operation of the system, it only affects the efficiency of clustering and only for the duration of a single epoch. Second, in our network model, there are no guaranteed direct and reliable links between nodes that contribute values and receivers, thus it is easy for a malicious node that forwards a proposed value to modify (if the value is not signed) or suppress that value, ensuring that a node never receives valid values and BA is never attained. Last, in our threat model, attackers are not interested in preventing consensus. They are only interested in achieving, with high probability, consensus on values selected by them.

Owing to the latter observation, our protocols impose restrictions on how values are contributed during the election. The most important restriction is that no party is able to predict the outcome of the agreement prior to committing to the value it plans to contribute.

We assume that the attacker has no compelling reason to prevent certain nodes from becoming aggregators, although such adversarial action would be plausible as a denial of service attack aiming to uneven load distribution and reduction of the network's lifetime.

IV. SANE Properties

We list the properties that should be satisfied by a practical and well-performing SANE (definition 3.1) protocol. Unless noted otherwise, we consider the listed properties required. With respect to *security*, a SANE protocol should attain:

- *Non-manipulability* - A party's contribution in the election process should not be able to influence the decision of honest nodes towards the election of selected nodes. We further call a protocol *strongly non-manipulable*, if in addition to the above property, no party has the ability to *prevent the election* of a targetted node. Unlike non-manipulability, strong non-manipulability is desirable but not required by a SANE protocol.
- *Authentication* - Each party should consider only the contributions of a restricted set of nodes. Contributing nodes should be capable of proving that they belong to this set. We note here that our protocol descriptions do not explicitly address authentication, however this is a property that can be efficiently achieved by incorporating existing WSN authentication solutions, e.g. [19].

- *Unpredictability* - An election protocol is predictable if the adversary can beforehand know the order in which nodes are elected as aggregators. This information could facilitate adversarial actions. For example, an adversary with the ability to compromise only a few nodes at a time, could use this knowledge to prevent a cluster of sensor nodes from transmitting information to its consumer during selected periods. Unpredictability is desirable but not required by a SANE protocol.

With respect to *correctness*, we consider the following properties:

- *Agreement* - In the absence of election contribution message losses, all parties belonging in the same *sector* should decide on the same alive node to become the next aggregator.
- *Adaptiveness* - In case election message losses occur, the protocol should not persistently yield partition of a sector into multiple clusters. Adaptiveness requirement is a desirable but not required property.
- *Node Fault Tolerance* - A single node's arbitrary failure should not result in failure of the election protocol. Node fault tolerance is a desirable but not required property.

Lastly, by *performance* we refer to the goal of extending the network's lifetime (Definition 2.2):

- *Load Balancing* - On average, all participating nodes should incur the same communication and processing load, such that energy consumption is flatly balanced.
- *Efficiency* - The election protocol should not introduce excessive communication overhead, which would counteract the benefits of load balancing.

V. Securing Randomized Aggregator Node Election

We describe three SANE protocols for achieving *secure*, *correct* and *well-performing* aggregator node election.

A. SANE based on Merkle's Puzzle and Homomorphic Encryption

The basic security building blocks of the first proposed approach are: i) an *additively homomorphic encryption transformation* [6]; and ii) a wireless-network-adapted configuration of *Merkle's puzzle* [18] (Appendix of [22]). We use a *homomorphic encryption transformation* for encrypting and summing up pseudorandom values contributed by all the nodes of the sector. We apply *Merkle's puzzle* for an initial simultaneous key agreement of $|N| - 1$ pairwise keys between the current aggregator node A_t and the remaining nodes in the sector, where $N \subseteq S$ is the set of functioning nodes in sector S . These keys are used by the sensor nodes to conceal their random values before they are distributed in the sector S . After a pre-defined and extremely short system time, the current aggregator A_t , at the end of epoch t , reveals the actual set of keys. In the following description we denote $E_k(v)$ the encryption of v under a key k by a

double additively homomorphic encryption scheme. This means that $E_{k+k'}(v+v') = E_k(v) + E_{k'}(v')$. The protocol works as follows:

1. At the end of epoch t the aggregator node A_t applies Merkle's puzzle to establish pairwise keys k_i with each of the sensor nodes $s_i \in N \setminus A_t$ and itself.

2. During the *encryption* phase, each sensor node $s_i \in N$, chooses a random value r_i and applies E . Subsequently, it computes the homomorphically encrypted (HE) sum $\sum_{j=1}^i E_{k_j}(r_j) = E_{k_i}(r_i) + \sum_{j=1}^{i-1} E_{k_j}(r_j)$, starting at s_1 with HE sum equal to $E_{k_1}(r_1)$. Node s_i adds itself to the list of nodes that have contributed to the sum, unicasts $\sum_{j=1}^i E_{k_j}(r_j)$ to s_{i+1} , and expects an acknowledgement from s_{i+1} . If s_{i+1} has failed, s_i reliably unicasts the message to the next available node. The process ends when all nodes including the aggregator A_t , have contributed and the sum $\sum_{j \in N} E_{k_j}(r_j)$ is available at all sensor nodes in N .

3. During the *decryption* phase the current aggregator node A_t floods the actual set of pairwise keys $\{k_j \mid s_j \in N\}$ and the ID of the node each key corresponds to, to all nodes in sector S . Note that these keys include the key with which A_t has homomorphically encrypted his contribution in step (2). Each node s_i individually decrypts the ciphered random sum $\sum_{j \mid s_j \in N_i} E_{k_j}(r_j)$ using the pairwise keys of nodes in the set N_i . $N_i \subseteq S$ consists of s_i and the rest of nodes in s_i 's sector that have contributed in the received HE sum. In case s_i does not receive the corresponding key k_j for node $s_j \in N_i$, it explicitly requests it from the aggregator. In case A_t persistently fails to deliver k_j to s_i , s_i proceeds with processing the encrypted sum without considering k_j . The ciphered random sum is decrypted by computing $k = \sum_{j: s_j \in N_i} k_j$ and subsequently performing the decryption operation $R_i = \sum_{j: s_j \in N_i} E_{k_j}(r_j) - k$.

The mapping function for converting the random aggregate value R_i to a node ID is defined as follows: 1) Each node s_i stores the node IDs of the nodes in the set N_i in an ordered set L , such that L^0 is the lowest ID node and $L^{|N_i|-1}$ is the highest; 2) Each node s_i elects its aggregator as $A_{t+1} = L^{R_i \bmod |N_i|}$.

Observation 5.1: Since the aggregator election process lasts only several seconds, the security provided by Merkle's puzzle only needs to hold for this duration.

Fact 5.2: For any $s_i, s_j \in N \setminus A_t$, no sensor node s_i can foresee the final result R_j at any honest node s_j prior to submitting its random value r_i . This is because it has no knowledge of the values of the contributions of the honest parties.

Lemma 5.3: Given the above Fact, the *non-manipulability* property (Section IV) is attained.

Proof 5.4: The Lemma 5.3 holds if the protocol combines node election contributions using a function that ensures that the contributions of adversarial nodes cannot

meaningfully influence the election outcome at honest nodes. This requires that the distributions of a *randomly* selected value x and the aggregate of a *randomly* selected value (in this case the value r_i contributed by an honest node s_i) with any other value (in this case the sum of the contributions of other nodes, here denoted y) are indistinguishable, i.e. $\{r_i \oplus y\} \equiv \{x\}$, where \oplus denotes the aggregation operation. Modular addition is such a function. Therefore, even if $|F_i| < |N_i|$ adversarial nodes collude into contributing values that are received by an honest node s_i and are combined with the values of honest nodes in $N_i \setminus F_i$, including the value of node s_i , the adversary would succeed in electing one of the adversarial nodes with probability no greater than $P_{suc} = |F_i|/|N_i|$. To ensure that an adversary cannot influence the election process of honest nodes, honest nodes need only include their contributed value in deriving their own random contribution sum. The security of the scheme is independent of the total number of nodes participating in the election and an attacker is only as powerful as its numbers.

Observation 5.5: The current aggregator node A_t can cheat in a meaningful manner only if it is aware of the final encrypted sum at any node s_i , $\sum_{j: j \in N_i} E_{k_j}(r_j)$. If we pre-configure a maximum allowed time Δt defined by the moment A_t floods the puzzle until it has to reveal the key set, one can sharply restrict the remaining time for a cheating aggregator node to modify keys in a meaningful way. Furthermore, a cheating aggregator node, which tries to reveal wrong keys aiming at manipulating the election process, is detected by at least one node.

Fact 5.6: The *agreement* property is attained. If all nodes retrieve the pairwise keys k_i from A_t of the nodes that have contributed in the HE sum they obtained, all nodes in a sector that have obtained the same HE sum, elect the same aggregator.

B. SANE Based on a Commitment Scheme

The second SANE protocol proceeds in two consecutive phases:

1. In the *commitment* phase, each sensor node s_i commits to a random value r_i , with $c(r_i)$, where $c(x)$ denotes a commitment on x that does not reveal x itself. Node s_i sends the commitment to his random value to all nodes in its sector.

2. In the *revealment* phase, during which nodes no longer accept commitment messages, the nodes send the actual random contributions r_i . Each receiving node s_i checks whether the commitment $c(r_j)$ from node s_j verifies for r_j . Subsequently, it sums up the random values sent by nodes in set N_i , $R_i = \sum_{j: s_j \in N_i} r_j$. $N_i \subseteq S$ consists of s_i and the nodes in S from which s_i has received random contributions that verified against their commitment.

The mapping function for converting the aggregated random value to a node ID is the same as the one for the

previous SANE protocol, except that list L now consists of s_i and the nodes from which a *valid* random value r_j has been received.

Fact 5.7: An attacker cannot pick a value r_j so that it fits its goals. By the time the attacker can send its encrypted random value, which is only during the *commitment* phase, it has not received any meaningful information. Also, during the *revelment* phase, the attacker cannot set r_j such that it favors him, since he has already committed to its initial r_j value in the elapsed *commitment* phase. In addition, given the commitment's security, it is hard for the attacker to derive a value r_j that is beneficial for him and still maps to the same $c(r_j)$ with the one he sent in the *commitment* phase.

Lemma 5.8: Given the above Fact, the *non-manipulability* property (Section IV) is attained.

Proof 5.9: We can prove Lemma 5.8 based on the same arguments used for Proof 5.4. To ensure that attackers cannot influence the election process of honest nodes, honest nodes need only include their contributed value in deriving their own sum R_i .

Fact 5.10: The *agreement* property is attained, since in the absence of $c(r_j)$ and r_j message losses, all nodes in a sector elect the same aggregator.

SANE based on Predetermined Random Values - A third SANE protocol reduces the communication overhead, at the cost of being *predictable*. In the previous schemes, random values are created such that nodes other than their generator do not know the pseudorandom generator seed. At this point, we observe that as long as the attacker releases its values at the same time with the honest nodes and is unable to modify its contribution at a latter stage, an attacker cannot control the output, even if the pseudorandom values are released in advance by the honest nodes.

Drawing from this observation, we use a function G , which generates a sequence of pseudorandom values of arbitrary length depending on a seed p . In other words once G is initialized with a seed p , it generates values $p^{(0)}, p^{(1)}, p^{(2)}, \dots, p^{(n)}$.

The scheme works as follows:

1. Prior to deployment all nodes in a sector S agree on a G .
2. In the *commitment* phase, each sensor node s_i randomly chooses a seed p_i . Subsequently, node s_i broadcasts p_i in its sector, so that each sensor knows the seeds of all nodes in its sector. By doing so, each node commits to a list of pseudorandom values $p_i^{(0)}, p_i^{(1)}, \dots$.
3. In round t , at each node s_i , $|N_i|$ nodes announce their availability and their values $p_1^{(t)}, \dots, p_{|N_i|}^{(t)}$ are treated as their random values. As the seeds and G are known, the random values of the available nodes can be computed by each sensor node independently.

Observation 5.11: Even if an attacker knows all the seeds that are chosen by the honest nodes *before* he has to

reveal his seed, finding a value which suits his goals would be practically infeasible. The reason is that his initial choice determines his contribution for each round without the possibility to influence it afterwards. Thus, an attacker would have to compute the list of pseudorandom values for each honest node and then select one seed which yields a list of values that, together with the other defined lists, fits his goals the best. If the number of rounds, for which the scheme is expected to be run, is not too small then this is a difficult task.

Lemma 5.12: Given the above Fact, the *non-manipulability* property (Section IV) is attained.

Proof 5.13: We can prove Lemma 5.8 based on the same arguments used for Proof 5.4. To ensure that attackers cannot influence the election process of honest nodes, honest nodes need only include the values generated by the function G using their *own seed* in deriving their random aggregate.

Fact 5.14: The *agreement* property is attained, since in the absence of availability announcement losses, all nodes in a sector elect the same aggregator.

Observation 5.15: This approach can be combined with the other two SANE protocols to reduce the communication overhead. Instead of invoking the other two protocols for each epoch, the contributions of the sensors in one epoch can be used as the seeds of G , and invoke this protocol instead.

VI. Comparison of Proposed SANE Protocols

In Section V, we derived that all three SANE protocols are equally capable of providing *non-manipulability* and *agreement*. Furthermore, due to the randomness in the election process, on average, all non-exhausted nodes in a sector are assigned the task of aggregator with the same frequency, thus all schemes are equally capable of providing *load balancing*. Below we provide in-depth comparison of the protocols with respect to the other properties listed in Section and IV. and the issue of *time synchronization*. Table I summarizes our observations.

Strong Non-Manipulability - In all schemes, the attacker is able to deny the election of targeted nodes, by suppressing messages intended for honest nodes. However, we consider these attacks difficult to launch undetected. Still, in the commitment-based and predetermined-randomness-based scheme the attacker *can inherently prevent* the election of a targeted node. The adversary can determine whether given the random node contributions or the availability announcements it has received so far and the contribution to which it has committed, a targeted node would be elected by the nodes that received the

	Priority	Merkle's puzzle	Commitment	Predetermined Randomness
Non-manipulability	Required	++	++	++
Strong Non-manipulability	Desirable	+	-	-
Unpredictability	Desirable	++	++	-
Agreement	Required	++	++	++
Adaptiveness	Desirable	+	++	++
Node Fault-Tolerance	Desirable	+	++	++
Load Balancing	Required	++	++	++
Efficiency	Required	+	+	++
Time Synchronization	Required	Inherent	Needs to be added	Needs to be added

TABLE I

COMPARISON OF THE PROPOSED SANE PROTOCOLS. +/- INDICATES TO WHICH DEGREE A PROPERTY IS FULFILLED BY THE PROTOCOL (MAXIMUM IS ++, MINIMUM IS -).

same messages as him. In this case, he would refrain from transmitting the random value it has committed to or its availability announcement. The Merkle's-puzzle-based scheme is not vulnerable to this type of attack. The attacker cannot predict the outcome of the random function before A_t reveals the individual node keys for the homomorphic encryption. This revelation takes place after all nodes have contributed their values. However, if the attacker controls the current aggregator it can prevent a node from being elected, by computing the expected random aggregate and refraining from releasing one of the HE pairwise keys k_i .

We note that our schemes do not satisfy or only partly satisfy the property of *strong non-manipulability*, as a trade-off for the property of *adaptiveness*. The reason is that there is no reliable way to differentiate between node or communication failures and an attacker that refrains from releasing a message during the election.

Unpredictability - Both the Merkle's-puzzle-based and random-commitment-based schemes are not *predictable*, as in every round newly released random values determine the election outcome at each node. In contrast the predetermined-randomness-based scheme is *predictable* in the absence of node failures, as the seeds and G are known and the random values of the available nodes can be computed beforehand independently.

Adaptiveness. In the Merkle-puzzle-based scheme, once a sector is partitioned in multiple clusters (due to failures during the propagation of the HE sum or persistent losses of messages carrying the pairwise keys k_i), it is not straightforward which of the multiple aggregator nodes will coordinate the Merkle puzzle in the next election. If no action is taken, a sector remains partitioned in subsequent elections. The other two protocols are *adaptive*, as they reach consensus once all the nodes in the sector start receiving the same flooded messages.

Node Fault-Tolerance - The Merkle's-puzzle-based scheme is the least *node-fault-tolerant*. It relies on a node with special functionality to coordinate the Merkle's-puzzle key distribution, thus the current aggregator node represents a single point of failure during the election. In addition, since the whole sector relies on each node to relay the homomorphically encrypted sum, the failure of the node that is currently computing the sum disrupts the process

and requires recovery steps. The other two protocols do not assign a special functionality to any node with during the election, thus they can tolerate the failure of any node.

Efficiency - The predetermined-randomness-based scheme is the most *efficient* of the three proposals in terms of communication overhead. It requires only one message per election process, which only needs to include the announcer's node ID. The commitment-based scheme requires two messages, which need to include a commitment c or a random value r . The commitment message should be large enough to maintain security of the commitment and $|r| \geq \lg N$, where N is the maximum number of nodes in a sector. The Merkle's-puzzle-based scheme requires that each node sends via unicast a message with the HE sum. It also requires that the current aggregator, floods messages for the Merkle's puzzle key distribution as well as the revelation of the pairwise decryption keys.

Time Synchronization - In the Merkle's-puzzle based protocol we rely on the current aggregator and coordinator of the Merkle's-puzzle to trigger the aggregator election phases. Hence, synchronization is inherent in this scheme. However, the other two schemes do not rely on the current aggregator to coordinate the next election process. Therefore, long-term clock synchronization is required. Low cost sensors are typically equipped with highly inaccurate clocks. Various techniques have been proposed for pair-wise and network-wide synchronization [13]. Our solutions are assumed to use a similar but simpler scheme that provides less accurate synchronization, while our election protocol takes into consideration synchrony inaccuracies.

VII. Additional SANE Design Issues

A. Message Authentication

All three proposed schemes require message authentication to prevent an attacker from assuming multiple identities in order to increase the probability of its election. For all protocols, prior to deployment, nodes in a sector are pre-configured with the set of nodes they should accept messages from. Nodes reject any message that cannot be attributed to the claimed originator.

Due to resource limitations, we do not use public key cryptography. to achieve message authenticity. Instead we

use μ Tesla [19]. Each node A produces a key hash chain, in which $K_0 = \text{hash}^n(K)$, where K_0 is pre-configured in all sensors in the sector and K is known only by A . The j th election protocol or data message sent by A in its sector is MACed with the key $K_j = \text{hash}^{n-j}(K)$. After sufficient time is allowed for all nodes to receive the message, K_j is revealed to all nodes in the sector. Then, each receiver node checks whether $\text{hash}^j(K_j) = K_0$ and if K_j verifies it uses it to check whether the authentication code of the j th message from node A is valid.

μ Tesla can be seamlessly integrated in all three schemes. However, all schemes need to incur additional communication overhead. Below we describe in detail how we augment the message flow of the proposed schemes to provide message authentication. For all protocols each node initializes a key hash chain of length n equal to the total number of messages it expects to send.

In the case of the Merkle-puzzle scheme, in step (1) (the Merkle puzzle phase), all broadcasted keys are MACed together with the μ Tesla key K_1 of the current aggregator. to produce an authentication code, which is verified in step (3). In step (3) the current aggregator reveals K_1 , while it authenticates the revealed pairwise keys $k_1, k_2 \dots$. NOTE: if a node does not receive all the broadcasted keys in the merkle puzzle and the encryption phase, the messages will not authenticate. So we would need to authenticate each message, which possibly contains only one key, independently. The j th homomorphically encrypted contribution is authenticated with the secret key K_j . Right after the contribution has been broadcasted to all nodes, while still in the *encryption* phase, the sensors send K_j in an additional message, which is not part of the initial election protocol.

In the case of the commitment-based scheme, a node creates a message authentication code to authenticate the commitment message in step (1) using the key K_j . He then piggy-backs K_j in the subsequent random value message. The key K_{j+1} for authenticating the random value message is sent in an additional separate message, while nodes are still in the *revealment* phase.

In the case of the pre-computed-randomness-based protocol, the initial seed and availability announcements are authenticated with K_j , which is revealed in a subsequent additional message.

B. Impact of Message Losses and Node Failures

Message losses and node failures can affect the proposed protocols, as they result in some nodes, during an election period t , to have different sets N of nodes in their sector that have sent valid messages. In particular, the discrepancy between the sets of nodes that have sent valid election protocol messages occurs as follows:

Merkle-puzzle-based protocol: In case the node that is next to receive the sum and add its contribution fails, the sum is

sent to the next node. In case a node fails while it computes the HE sum, the so far computed sum is lost and the process needs to be restarted.

Nodes transmit the homomorphically encrypted (HE) sums reliably, thus message losses are handled by the protocol. Some node's may not receive some of the homomorphic encryption keys k_i , which are intra-sector flooded by the current aggregator. In that case, they would explicitly "pull" the keys from the aggregator. In case the aggregator persistently fails to deliver k_i , k_i is not considered in decrypting the random aggregate and possibly elect a different aggregator than the nodes that received s_i 's contribution. In such case, our protocol still functions correctly, as a node may function both as an aggregator for some nodes in its sector, and as sensor that has elected another node as its aggregator. Therefore, packet losses in the worst case result to less efficient clustering, but do not affect the correctness of the protocol.

Commitment-based protocol: Some nodes within a sector may not receive all the valid random values transmitted by the nodes in the sector. This would result in nodes within the same sector possibly electing different aggregators. As above, the protocol would still function correctly.

Precomputed-randomness-based protocol: some nodes may not receive another node's availability announcement. This would result in nodes within the same sector possibly electing different aggregators. As above, the protocol would still function correctly.

Frame losses may occur due to collisions or due to failed (e.g. exhausted) forwarding nodes. We reduce the number of MAC layer collisions by ensuring that no two nodes start transmitting their messages simultaneously within a phase. This is achieved by having nodes backing off a random short period before they start transmitting.

C. Time Synchronization

In the Merkle's-puzzle based protocol we rely on the current aggregator and coordinator of the Merkle's-puzzle to trigger the aggregator election phases. Hence, synchronization is inherent in this scheme.

The commitment-based protocol does not rely on the current aggregator to coordinate the next election process. Therefore, long-term clock synchronization is required. However we need a secure decentralized time-synchronization mechanism because an attacker may attempt to change the global time so that it extends the time it remains aggregator. Low cost sensors are typically equipped with highly inaccurate clocks. Various techniques have been proposed for pair-wise and network-wide synchronization [13]. Most recently, [23] Sun et al. introduced a secure network-wide synchronization technique (Tiny-SerSync) that builds on authenticated single-hop pairwise synchronization and on the μ Tesla broadcast authentication protocol. Our solution can be used in conjunction with

TinySerSync, so that a single trusted node in the network or per sector can maintain global synchrony, by periodically sending synchronization pulses.

VIII. Experimental Evaluation

A. Simulation

The purpose of our simulation is to demonstrate that in some typical WSN deployment scenarios, randomized selection of aggregator nodes yields satisfactorily balanced energy distribution when compared to residual-energy-aware election protocols. Our intuition is that at the early stages of a network's life the energy imbalances caused by the fact that, certain nodes may incur higher communication load than others, do not have a profound impact on the network's lifetime. The energy imbalances only affect the network towards the end of its lifetime when all nodes are close to exhaustion, and a slight difference in energy levels makes the difference between a node's life or death. We also compare our approach to a degree-based aggregator election scheme.

1) *Simulation Environment and Energy Consumption Model:* We use the Glomosim [26] wireless network simulator. We set the node parameters to closely match the specification of a Crossbow MICAz Mote [1]. The radio operates at 2.4 GHz, its maximum transmission range at 70-100 m outdoors (line of sight) at 250 Kbps.

Since IEEE 802.15.4/Zigbee is not included in the Glomosim distribution, we use Glomosim's IEEE 802.11 MAC layer. The main difference between the two MAC schemes is that Zigbees has exhibits higher power efficiency and has reduced transfer rates, thus we only need to follow an appropriate radio energy model and set 802.11 transmission rate much lower than it usually supports. In addition we consider that 802.11 has 464 bit frame control overhead, while 802.15.4's is only 120 bits for short addresses. 802.15.4's data payload size is at most 127 bytes, so in order to simulate correctly an 802.15.4 frame with x byte payload (where $x \leq 127$), we send an 802.11 frame with $x - 43$ byte payload. For unicast transmissions, our MAC always employs virtual carrier sensing (VCS) VCS is disabled for broadcast transmissions.

We adopt the simple radio energy dissipation model used in [14, 21, 29]. This model assumes $1/d^n$ path loss, in which n is dependent on d . In order to transmit k bits data at distance d , the energy spent by the transceiver is $E_T(k, d) = k(E_{elec} + E_{amp})d^n$. E_{elec} is the electronics parameter and depends on factors such as the digital coding, modulation, filtering, and spreading of the signal, while E_{amp} is the amplifier parameter, which depends on the distance to the receiver and the acceptable bit-error rate. E_{elec} is set equal to 50 nJ/bit. When $d < d_0$, we assume a free space propagation model under which $E_{amp}=0.01$ nJ/(bit m^2) and $n=2$. When $d \geq d_0$, we assume a multipath fading propagation model under which $E_{amp}=0.0013$ pJ/(bit m^4) and $n = 4$. The value d_0 is the threshold distance,

which depends on the environment and is commonly set equal to 75 m for line-of-sight, outdoors environments. We implement the communication energy model in the 802.11 module, so that we account for all received and transmitted bits by a node (e.g headers of frames not destined for the node and carrier sense operations). Similar to [14], we model the energy dissipated for data aggregation as 5 nJ/bit/signal. In all experiments, we set each node's initial energy equal to 1 Joule.

2) *Deployment Scenario:* We simulate the deployment of 500 sensors over a 100m x 100m area. We use the following simple application scenario. The network forms a grid of 25 20x20 m^2 sectors, which contain 20 nodes each. The placement of nodes within a sector is random. To prevent collisions due to synchronization, nodes always wait a short random amount of time before they transmit any message.

We use a simple reverse-gradient-based routing technique to establish routes between sensor nodes within a sector. Upon initialization, a node floods its sector with *route-announcements*. Sensors rebroadcast first-seen *route-announcements* and update their next hop to the sender of the announcement to be the node that relayed the first-seen *route-announcement* to them. Messages from sensors nodes to the aggregator are forwarded along the established routes. Nodes unicast messages to the next hop to the destination with transmission range equal to 15 m. Periodically, all nodes send intra-sector route announcements to ensure that connectivity is maintained in the presence of node failures.

To illustrate the significance of fairly assigning the aggregator task, aggregators perform particularly expensive operations. They send data to a sink node residing in the center of the area by directly unicasting them with transmission range equal to 75 m. Note that for larger areas, aggregators may need to re-broadcast another aggregator's first-seen message to ensure that this message reaches the sink.

3) *Random versus Rule-based Aggregator Election:* We compare our approach to probabilistic *residual-energy-based* and *degree-based* solutions. We do not implement an election protocol such as LEACH or DCA, but we simulate the behavior of the network assuming that a mechanism exists to centrally elect a node in their sector according to a weighted probability. The weights considered are the residual energy and the number of one hop neighbors (within 15 m range). We compare them to uniformly random election of a node in a sector. The probability p_i of election of a node i which has weight w (residual energy or degree) is computed as $p_i = w_i / \sum_{i \in S} w_i$

In each sector, every T_{agg} , sensor nodes send frames to the aggregator with 120 byte payload plus 120 bit frame 802.15.4 MAC and physical header, using multi-hop forwarding with transmission range d equal to 15 m. According to our model, each such message transmission and reception costs 56430 and 54000 nJ respectively. Every T_{elect} , sensor nodes elect a new aggregator. Sensors within

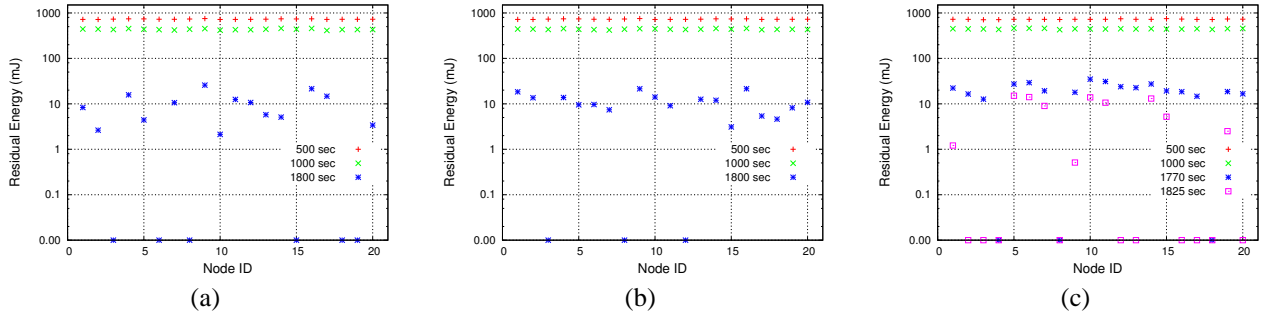


Fig. 1. Snapshots of per-node residual energy: (a) residual node energy for random aggregator node election; (b) residual node energy for energy-based aggregator node election; and (c) residual energy for degree-based aggregator node election. Nodes 1-20 belong to a sector residing close to the center of the 100x100 area. Although the y axis is logarithmically scaled, it is adjusted such that it depicts 0 J residual energy values.

the sector forward the data to the aggregator. Sensor and aggregator nodes are active only during the aggregation phase. Sensor nodes forward messages to the aggregator

The aggregator fuses the received data. We assume that the aggregator fuses the complete received 120 byte message, thus the computation energy cost for each message is 4800 nJ. Subsequently, it transmits a single 120 byte payload message with range 75 m, to forward traffic to a sink. This transmission costs 98423 nJ. Hence, in each aggregation phase the aggregator incurs the highest message reception, transmission and computation overhead.

In the first experiment, we select aggregators randomly. In the second experiment, we select aggregators based on their energy (biasing the election towards nodes with the highest residual energy). Nodes do not use an election protocol instead the simulation elects them according to their residual energy. In the third experiment, we elect the aggregator node based on the number of its one-hop neighbors (node degree). This criterion ensures that the aggregator is highly connected, meaning that it is less likely that messages intended for it from the sensor nodes would be lost. It also means that, on average, messages require less hops to reach it. T_{agg} is always equal to 1 sec. We set T_{elect} equal to 5 sec. The simulation virtual time of both experiments is 2000 seconds.

By analyzing our simulation logs, we observe that at the early stages of the network's life, sensor nodes expend roughly $400 \mu J$ per aggregation phase (per second), while aggregators expend roughly $1400 \mu J$. In the randomized protocol, nodes become aggregators on average once every 20 elections, that is 100 seconds. In the degree-based approach, nodes have 15-19 one-hop neighbors in their sector and usually 2-4 only highly connected nodes have 19 such neighbors. These highly connected nodes are on average elected approximately every 80-90 seconds while the rest about every 110-125 seconds.

We evaluate the aggregator election schemes for the cases the nodes have high and very low initial energy. In Figure 1, we plot graphs of how energy is distributed among the sensors during the initial non-critical phase of the network and during the critical phase, when many nodes are close to exhaustion. We observe that in the early

phases of the network's life (500-1000 sec) the energy levels of all nodes are almost equally balanced in both the randomized and the energy-based election. At the late phases of the networks lifetime, approximately after 1800 seconds, we observe the following about the state of the nodes: a) in the random election, 6 out of 20 nodes are completely energy depleted; b) in the energy-based election, 3 out of the 20 nodes are depleted. This does not signify a substantial difference with respect to network lifetime, especially when the network lifetime is defined according to connectivity (see Definition 2.2). It is very likely that sufficient connectivity between the alive nodes in a sector is maintained with both 17 or 14 nodes being active.

In the degree-based election, we observe that at the early stages the energy levels of the three nodes (4, 8 and 18), with the most neighbors (18-20) deviate visibly from the other nodes that have less one-hop neighbors (12-15). As a result, at 1770 secs, these three nodes are already exhausted, whereas only two nodes are exhausted in the random and no nodes in the energy-based election.

These results indicate that for common application scenarios similar to the one we simulate, random election does not yield highly uneven energy consumption and thus does not reduce the network lifetime. We observe however, that random election is less suitable than the residual-energy-based approach when network energy is close to exhaustion. Degree-based approaches result to highly connected nodes to be exhausted faster. Consequently, the network loses substantial connectivity in its early life stages. At the early stages of the network's life, we have a dense network, therefore connectivity is ensured even though it may not be optimal. However, with energy based or random based aggregator election, we ensure that lifetime of nodes is extended (in comparison with other approaches) and we avoid a situation where nodes that are important to connectivity die too early. At the latter network life stages, using a particular aggregator election technique should not yield any gains since connectivity is largely lost.

IX. Related Work

We classify prior work on cluster head (CH) election protocols into the following categories: a) fully randomized; b) probabilistic residual-energy-based [14, 29]; c) degree-based [8, 16] d) connectivity-based for energy-efficient routing [7, 28]; e) ID-based [3, 4, 12]; f) generic weight-based [5]; g) geographically-aware [10, 27]; and h) voting- and residual-energy-based [21].

Fully Randomized Wang et al. [24] proposed a randomized election protocol for cluster formation in semi-mobile ad-hoc networks. Their scheme is reminiscent of our commitment-based scheme, i.e. nodes first transmit a hash of their random values and subsequently they transmit the random value. However, their protocol has a higher communication overhead, as they assume a semi-dynamic network topology (it is static during an election protocol run) and does not consider the resource scarcity of WSNs.

Probabilistic Residual-energy-based LEACH [14] follows a probabilistic approach. In its energy-aware variation, each node independently and probabilistically decides whether to become aggregator according to its residual energy. It aims at optimizing network lifetime by equally distributing energy consumption. Nodes make autonomous decisions on whether they will announce themselves cluster heads or join existing clusters.

HEED [29] is also a probabilistic energy aware protocol. It assumes that nodes have multiple transmission levels and that nodes within a cluster have one hop distance from the aggregator. HEED can be generalized with sensors residing multiple hops away from the aggregators and aggregators residing multiple hops from each other. The primary aggregator election criterion is the residual energy. The secondary criterion, used to break ties, is the cost defined as either the node degree or the mean of the minimum power needed by the nodes in the cluster to reach the CH or the inverse of the node degree. HEED extends LEACH by incorporating communication range limits and cost information.

Degree-based Protocols. Kuhn et al.'s [16] scheme creates minimum connected dominating sets by computing an asymptotically optimal clustering in polylogarithmic time. It can operate in the absence of any MAC layer. They discuss a protocol for setting up periodic sleep/listen schedules within clusters in a quick and energy-efficient way, which is based on their proposed clustering algorithm. Their goal is to derive efficient transmission schedules between nodes and CHs to avoid collisions and loss of connectivity. With respect to our design space, the main pitfalls of this approach are: a) it tends to favor aggregators with high degree, forming dense clusters and resulting in quick battery drainage of highly connected nodes; and b) it assumes that there are 3 independent communication channels available.

Chan et al. proposed ACE [8], which favors nodes with high degree in electing aggregators and it has a constant cost. If a node n thinks it has the highest degree, it announces to its neighbors the intention to become aggregator. Receivers of the announcement become node n 's followers.

In the next election round, the CH decides which of its followers would have the greatest number of followers and delegates the CH role to it. Since it favors nodes with high degree, it results to uneven load distribution. In addition, from the security perspective it is not desirable to have the current cluster head deciding on which node will be the next CH.

Connectivity-aware Protocols for Routing PEAS [28] builds a long-lived sensor network and maintains robust operations using large quantities of short-lived sensor nodes. PEAS extends system lifetime by keeping only a necessary set of sensors working and putting the rest into sleep mode. Sleeping nodes occasionally wake up to probe the local environment and replace failed ones. The sleeping periods are self-adjusted dynamically, so as to keep the sensors wake-up rate roughly constant, thus adapting to high node densities.

ASCENT [7] builds on the notion that as density increases, only a subset of the nodes are necessary to establish a forwarding backbone. In ASCENT, each node assesses its connectivity and adapts its participation in the multi-hop network topology. This system achieves linear increase in energy savings as a function of the density and the convergence time required in case of node failures, while it maintains adequate connectivity.

Generic-weight-based DCA [5] builds a dominating set of high weight nodes. The weight may depend on node degree, residual energy and other metrics. DCA is an iterative technique with a relatively high $O(Diameter)$ cost. Before making a decision, a node waits for his one-hop neighbors with higher weights to decide on whether to become aggregators or to join existing clusters. Upon receiving one or more cluster head announcements a node n decides to follow the CH with the larger weight. If all nodes with larger weight in n 's neighborhood do not send a cluster head announcement message but instead announce that they join another cluster, n can announce itself as a CH. The CH election algorithm is deterministic and depends only on the weights. Similar to the above techniques this algorithm relies on nodes reporting their true weight and not electing themselves unless it is appropriate. In this way it allows an attacker to manipulate the election decision.

Location-based GAF [27], similar to SPAN [10], exploits node redundancy to form energy efficient network topologies. GAF selects particular nodes to participate in tasks such as sensing or forwarding, while the other nodes in the same region are inactive. In GAF, a region is the area A in which any node can communicate with any other node V in area B, where B is neighboring to A. Thus only one node in region A needs to be forwarding messages to region B. SPAN generalizes the region based on 2 hop connectivity. GAF and SPAN assume that the position of nodes and the field dimension is known. Both protocols focus on the role of CH's as forwarders not as aggregators, thus their optimizations are tuned for this purpose. For example, these protocols strive to determine equivalent

nodes, i.e. nodes that can provide equivalent forwarding capabilities in order to keep only one of those nodes active.

ID-based Baker et al.'s *Linked Cluster Algorithm* [4], LCA, uses TDMA to communicate election messages with all nodes in a network. A node n becomes head of a cluster if at least one of the following conditions is satisfied: (a) n has the highest identity among all nodes within one hop from it, (b) x does not have the highest identity in its 1-hop neighborhood, but there exists at least one neighboring node y such that x is the highest identity node in y 's 1-hop neighborhood. The LCA heuristic was revised in [12] to decrease the number of cluster heads produced in the original LCA. LCA was intended for small networks of less than 100 nodes. In such networks the time between transmissions from a node is short and can be tolerated. However, for microsensor networks of hundreds of nodes, LCA would incur great delays between node transmissions in the TDMA communication scheme. In addition, this scheme heavily relies on global clock synchronization, which is not available in a large low cost microsensor networks. With respect to network lifetime, this scheme tends to favor nodes with specific IDs, resulting in those nodes incurring disproportionate energy consumption.

In [3], the authors show that the minimum d -hop dominating set problem is NP-complete. They present a heuristic to form d -hop clusters in a wireless ad-hoc network, in which nodes have a deterministic mobility pattern. When the heuristic terminates, a node either becomes a cluster head, or is at most d wireless hops away from its cluster head. The number of messages sent between nodes has constant cost $O(d)$. It minimizes the number of cluster heads and it fairly distributes load among cluster heads. Its main drawback with respect to its use in sensor networks is that it tends to re-elect nodes as cluster heads, when the network configuration remains largely unchanged in order to avoid the cluster head transition overhead.

Voting-based In [21], the authors propose the Voting-based Clustering Algorithm (VCA) for data dissemination in quasi-stationary sensor networks. Their approach lets sensors vote for their neighbors to elect suitable cluster heads. VCA combines load balancing, energy and topology information in a simple voting scheme. The protocol is completely distributed, it does not make any assumptions about sensor location and network topology. VCA can reduce the number of clusters by 5-25% and prolong the lifetime of a sensor network by 25-30% over that of existing energy-aware clustering protocols, such as LEACH and HEED. VCA addresses inefficient cluster formation by enabling nodes to exchange information about their local network view, through a voting scheme. Their scheme picks the weight of the vote for a neighbor node A received by another node, to be inversely proportional to the degree and proportional to the residual energy of node A . The weakness of this scheme is that it relies on nodes faithfully reporting their degree and residual energy.

Secure Aggregation for Wireless Networks Hu et

al. [15] propose a scheme that preserves the integrity of sensor network information during its aggregation, while being forwarded to the sink, in the presence of malicious aggregators. This scheme does not prevent the election of compromised nodes as aggregators, thus it does not deal with eavesdropping and the sink-hole attack, which is explicitly addressed by our proposal.

SIA [20] addresses the problem of secure information aggregation with respect to integrity. The scheme by Chan et. al [9] also detects any manipulation of the aggregate by the arbitrary numerous adversaries beyond what is achievable through data contribution of compromised nodes. Their solution is suitable for general hierarchical topologies in which nodes may have multiple aggregators, whereas SIA is designed for single-aggregator topologies.

Corsby et. al [11] propose using a reputation mechanism based on the observations of promiscuous receivers. They rely on the premise that malicious nodes will have a distinct communication pattern (e.g. not forwarding messages), that would enable compliant nodes to identify them. However, such an assumption is not always applicable.

X. Conclusion

We have provided and compared two practical and secure aggregator election protocols for WSNs. These protocols are capable of ensuring election non-manipulability, while fulfilling several desirable properties with respect to *correctness* and *performance*.

References

- [1] The micaz crossbow mote. available at www.xbow.com.
- [2] M. Acharya, J. Girao, and D. Westhoff. Secure comparison of encrypted data in wireless sensor networks. In *WiOpt*, 2005.
- [3] A. D. Amis, R. Prakash, D. Huynh, and T. Vuong. Max-min d -cluster formation in wireless ad hoc networks. In *INFOCOM*, 2000.
- [4] D. J. Baker and A. Ephremides. The architectural organization of a mobile radio network via a distributed algorithm. In *IEEE TC*, 1981.
- [5] S. Basagni. Distributed clustering for ad hoc networks. In *I-SPAN*, 1999.
- [6] G. T. C. Castelluccia, E. Mykletun. Efficient aggregation of encrypted data in wireless sensor networks. In *MOBIQUITOUS*, 2005.
- [7] A. Cerpa and D. Estrin. Ascent: Adaptive self-configuring sensor networks topologies. In *IEEE TMC*, 2004.
- [8] H. Chan and A. Perrig. ACE: An emergent algorithm for highly uniform cluster formation. In *EWSN*, 2004.
- [9] H. Chan, A. Perrig, and D. Song. Secure hierarchical in-network aggregation in sensor networks. In *ACM CCS*, 2006.
- [10] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *MOBICOM*, 2001.
- [11] G. V. Crosby, N. Pissinou, and J. Gadze. A framework for trust-based cluster head election in wireless sensor networks. In *DSSSSNS*, 2006.
- [12] A. Ephremides, J. E. Wieselthier, and D. J. Baker. Design concept for reliable mobile radio networks with frequency hopping signaling. In *Proceedings of IEEE*, 1987.
- [13] Ganeriwal. Tpsn. In *Sensys*, 2003.
- [14] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. In *IEEE TWC*, 2002.
- [15] L. Hu and D. Evans. Secure aggregation for wireless networks. In *WSAAN*, 2003.
- [16] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Initializing newly deployed ad hoc and sensor networks. In *MOBICOM*, 2004.
- [17] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. In *ACM TPLS*, 1982.
- [18] R. C. Merkle. Secure communications over insecure channels. In *Communications of the ACM*, 1978.

- [19] A. Perrig, R. Szewczyk, V. Wen, D. E. Culler, and J. D. Tygar. SPINS: security protocols for sensor networks. In *WINE*, 2001.
- [20] B. Przydatek, D. Song, and A. Perrig. Sia: Secure information aggregation in sensor networks. In *ACM SenSys*, 2003.
- [21] M. Qin and R. Zimmermann. An energy-efficient voting-based clustering algorithm for sensor networks. In *SNPD*, 2005.
- [22] M. Sirivianos, D. Westhoff, F. Armknecht, and J. Girao. Non-manipulable aggregator node election in wireless sensor networks. www.ics.uci.edu/~msirivia/sane.pdf, 2006.
- [23] K. Sun, P. Ning, and C. Wang. Tinsersync: secure and resilient time synchronization in wireless sensor networks. In *CCS*, pages 264–277, 2006.
- [24] X. Wang, X. Guo, and X. Yin. Nrcr: A new random cluster election algorithm. In *Journal of Communication and Computer*.
- [25] D. Westhoff, J. Girao, and M. Acharya. Concealed data aggregation for reverse multicast traffic in sensor networks: Encryption, key distribution, and routing adaptation. In *IEEE TMC*, 2006.
- [26] Z. Xiang, RajiveBagrodia, and MarioGerla. Glomosim: A library for parallel simulation of large-scale wireless networks. In *PADS Workshop*, 1998.
- [27] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *MOBICOM*, 2001.
- [28] F. Ye, G. Zhong, J. Cheng, S. Lu, and L. Zhang. Peas: A robust energy conserving protocol for long-lived sensor networks. In *ICDCS*, 2003.
- [29] O. Younis and S. Fahmy. HEED: A hybrid, energy-efficient, distributed clustering approach for ad-hoc sensor networks. In *INFOCOM*, 2004.

APPENDIX

MERKLE'S PUZZLE

Merkle's puzzle is a key exchange protocol dating back to 1978 [18]. The basic idea can be seen as a mixture of cryptography and steganography. We denote the two parties by A and B , with A being the initiator of the key exchange. In a first step, A generates many different keys K_i and corresponding identifiers ID_i . It encrypts each tuple (K_i, ID_i) with a small random key and sends the whole set of ciphertexts to B . B randomly selects one of the keys by picking one ciphertext and breaking it. This is possible as the keys used for the encryption are considered to be small. To tell A which key is the chosen one, B sends the corresponding identifier to A . Thus, A can simply look up which key is connected to the identifier it received to determine the key. From now on, this key serves as a shared secret between A and B . An eavesdropper gets in the first hand only knowledge of the ciphertexts and the identifier. As he doesn't know to which ciphertext the identifier belongs, the only approach is to break ciphertexts until he finds the correct one. This means that an adversary has to break on average half of the amount of ciphertexts.

DOUBLE HOMOMORPHIC ENCRYPTION SCHEME

Consider a mapping ϕ between two sets (S, \circ) and $(T, *)$ with \circ and $*$ being binary operations, that is $s \circ s' \in S$ and $t * t' \in T$. ϕ is called homomorphic if it holds that

$$\phi(s \circ s') = \phi(s_1) * \phi(s')$$

for all $s, s' \in S$. This means that the structure of (S, \circ) is preserved under the mapping to $(T, *)$. A homomorphic encryption scheme is a an encryption algorithm $E_k(v)$ which is homomorphic in k and/or in v . An example is the RSA-scheme $x \mapsto x^e \pmod n$. Because of $(x \cdot y)^e = x^e \cdot y^e$, RSA is homomorphic in the plaintext.

A double homomorphic encryption scheme is an encryption that is homomorphic in both the key and the plaintext. That it it holds that

$$E_k(v) \bullet E_{k'}(v') = E_{k \circ k'}(v * v')$$

with $\bullet, *, \circ$ being binary operations on the sets of ciphertexts, plaintexts, and keys, respectively. In general, a homomorphic property of an encryption scheme is rather considered as a weakness. The reason is that it reveals more structural information between plaintexts and ciphertexts compared to non-homomorphic schemes. However, for several purposes this property serves well for certain tasks. Examples are electronic voting or concealed data aggregation.