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 information produced by a Wireless Sensor Network (WSN). Despite this fact, existing aggregator node election schemes do not address an adversary that strives to influence the election process towards candidate nodes 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 and compare three protocols that instantiate such function.**

I. Introduction

Cluster head node election protocols such as LEACH [12], HEED [25] and VCA [18], 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 incented 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 [1, 5, 21] and integrity [8, 17] 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 three secure aggregator 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.

The rest of this paper is organized as follows. Section II introduces randomized aggregator node election, and compares it to residual-energy- and connectivity-based approaches. Section III describes our assumptions with respect to the network and the threat model, and defines a Secure Aggregator Node Election Protocol (SANE). Section IV discusses the properties that a SANE protocol should satisfy. Section V describes the proposed SANE protocols and Section VI compares them. Section VII discusses additional design issues. In Section VIII, we comparatively evaluate randomized and weight-based node election. We discuss related work and conclude in Sections IX and X, respectively.

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 nodes that belong to a predefined and finite set S of microsensor nodes. Nodes in set S use this algorithm in order 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.

Definition 2.2: The *network lifetime* of a WSN is the latest epoch in which the set of alive nodes (i.e. nodes that have not failed) can effectively communicate the information produced by the microsensor network to its consumer.

An aggregator node election protocol in microsensor networks strives to flatly balance the energy consumption of the network, while maintaining connectivity between the sensor nodes and the consumer of the WSN's information. It achieves 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, under which nodes with higher remaining energy are more likely to be elected as the aggregator nodes [4, 12, 18, 25]; b) protocols in which nodes with high connectivity (i.e. many one-hop

This work is supported by the European Commission within the STReP UbiSec&Sens of the EU Framework Program 6 for Research and Development (IST-2004-2.4.3). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the UbiSec&Sens project (www.ist-ubisecsens.org) or the European Commission.

neighbors) are favored to be the aggregator node [2, 7, 13]. 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 alive (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 alive (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 Section VIII, we defend these statements based on 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

A. Network Model

We assume a fully decentralized network of equal functionality and capability microsensor 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.

We further assume a CSMA/CA MAC scheme (such as IEEE 802.15.4) for broadcast and unicast communication. 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.

B. Threat Model

The adversary can eavesdrop 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.

Definition 3.1: An adversary is *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 $P_{suc} = |F_i|/|N_i|$.

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 means for causing uneven load distribution and reduction of the network's lifetime.

C. Objective

At this point, we revisit the definition of an aggregator node election protocol. Our objective is an aggregator node election protocol that addresses the network and threat model presented in Sections III-A and III-B.

Definition 3.2: A *Secure Aggregator Node Election (SANE)* protocol is a distributed algorithm running on 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, that have received the same set of valid election contributions for epoch t, reach consensus

in electing a particular node $A_t \in S$. Their election decisions should *not be manipulated* towards the election of any node.

It may appear that a SANE protocol solves a flavor of the Byzantine Agreement (BA) problem [14]. 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 the value proposed by a participating party. If multiple parties propose values, honest nodes must agree on the same set of proposed values in order to derive a common value.

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. Firstly, 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. Secondly, 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. Lastly, 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.

IV. SANE Properties

We list the properties that should be satisfied by a practical and well-performing SANE (Definition 3.2) 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 preselected 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 preselected 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. the ones proposed in [16].
- *Unpredictability* - An election protocol is predictable if the adversary can aforehand 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 that belong 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 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, nodes in a sector should incur the same communication and processing load, such that energy consumption in the network 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 the required properties for *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* [5]; and ii) a wireless-network-adapted configuration of *Merkle's puzzle* [15]. In the Appendix, we provide a brief description of both security primitives.

We use the 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 predefined 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 $\{k_j : s_j \in N\}$ 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 $\sum_{j:s_j \in N_i} E_{k_j}(r_j)$ 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 $A_{t+1} = L^{R_i \mod |N_i|}$.

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 Fact 5.2, the *non-manipulability* property (Section IV) is attained.

Proof 5.4: 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 preconfigure 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 *revealment* 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 Fact 5.7, 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.

C. 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 $p^{(0)}, p^{(1)}, p^{(2)}, \ldots p^{(n)}$.

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 $p_i^{(0)}, p_i^{(1)}, \ldots$

3. In round t, alive nodes in the sector broadcast a short message to announce their availability. At each node s_i, the values $p_1^{(t)}, \ldots, p_{|N_i|}^{(t)}$ for the nodes from which s_i has received availability announcements are treated as the random contributions of the nodes in N_i. As the seeds and G are known, the random values of the available nodes can be computed by each sensor node independently.

The mapping function for converting the aggregated random value to a node ID is the same as the one for the previous SANE protocols, except that list L now consists of s_i and the nodes from which an availability announcement has been received.

Fact 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, this is a difficult task.

Lemma 5.12: Given Fact 5.11, 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, all alive nodes in a sector are assigned the energy consuming aggregator task with roughly the same frequency, thus all schemes are equally capable of providing *load balancing*. In this section, we provide in-depth comparison of the protocols with respect to the other properties listed in Section IV and the issue of *time synchronization*. Table I summarizes the following observations:

- *Strong Non-Manipulability* - In all schemes, the attacker is able to deny the election of targeted nodes, by suppressing messages sent to honest nodes. the HE sum to the targeted node, However, we consider these attacks difficult to launch undetected. In the commitment-based and predetermined-randomness-based scheme the attacker *can prevent* the election of a targetted node even without suppressing sent messages. The adversary can determine whether given the random node contributions or the availability announcements it has received so far and the contribution to which the adversary has committed, the targeted node would be elected by the nodes that received the 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. The revealment 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.
- *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

	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 -).

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 consencus 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 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 $|r| \geq N$, 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 revealment 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. We discuss the issue of synchronization in Section VII-B.

VII. Further SANE Design Issues

In this section we discuss the impact of message losses and node failures on the operation of the proposed protocols as well as the issue of time synchronization.

A. Impact of Message Losses and Node Failures

Message losses and node failures can affect the proposed election protocols, as they result in some nodes, 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 may occur as follows:

• *Merkle-puzzle-based protocol* - In case the node that is scheduled to receive the sum and add its contribution fails, the sum is sent to the next available 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 using acknowledgments and retransmissions, 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. This is because 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 result to less efficient clustering, in the worst case, 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.

B. 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- and predetermined-randomness-based protocols, 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. In addition, 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. Various techniques have been proposed for pairwise and network-wide synchronization [11]. Recently, Sun et al. [19] introduced TinySerSync, a secure network-wide synchronization technique that builds on authenticated single-hop pairwise synchronization and on the μTesla broadcast authentication protocol. Our solution can be used in conjunction with TinySerSync, so that a single trusted node in the whole network or in a sector can maintain global synchrony, by periodically sending synchronization pulses.

VIII. Experimental Evaluation

The purpose of our simulation-based evaluation is to demonstrate that in a typical WSN deployment scenario, randomized selection of aggregator nodes yields satisfactorily balanced energy distribution, when compared to residual-energy- and connectivity-based election protocols. Our intuition is that at the early stages of a networks life the communication load inbalances do not have a profound impact on the network's lifetime. The load imbalances would 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.

A. Simulation Environment and Energy Consumption Model

We use the Glomosim [22] wireless network simulator. The radio operates at 2.4 GHz, its maximum transmission range is set at 75 m and its transfer rate 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 schemes is that Zigbee has higher power efficiency and has reduced transfer rates. Thus, we 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 control overhead is 464 bits per frame, while 802.15.4's is only 120 bits for short addresses. Therefore, 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, the simulated MAC employs virtual carrier sensing (VCS), while VCS is disabled for broadcast transmissions.

We adopt the simple radio energy dissipation model used in [12, 18, 25]. This model assumes $1/d^n$ path loss. 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 we set equal to 75m, as commonly done for line-of-sight environments. Similar to [12], we model the energy consumption for data aggregation as 5 nJ/bit/signal. In all experiments, we set each node's initial energy equal to 1 Joule.

B. Deployment Scenario

We simulate the deployment of 500 sensors over a 100m x 100m area. 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.

Sensor nodes use reverse-path routing to route messages to the current aggregator. Upon initialization, each sensor 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. 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 substantially more expensive operations than sensor nodes.

C. Random versus Weight-based Aggregator Election

We compare our randomized approach to probabilistic *residual-energy*- and *connectivity-based* solutions. The simulation centrally elects a node to become aggregator of a sector according to a weighted probability. The weights considered are the residual energy and the number of one hop neighbors (within 15 m range). The probability p_i with which a node i with weight w (residual energy or connectivity) is elected, is computed as $p_i = w_i / \sum_{i \in S} w_i$.

The simulation virtual time of the experiments is 2000 seconds. A new aggregator is elected every 5 seconds. In the first experiment, the aggregator is elected randomly. In the second experiment, the aggregator is elected according to the above mentioned probability p_i, using the residual energy of each node as the weight w_i. In the third experiment, the aggregator is elected according to probability p_i and we use the number of one-hop neighbors (node degree or connectivity) of each node as the weight w_i.

In each sector, every 1 second, 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

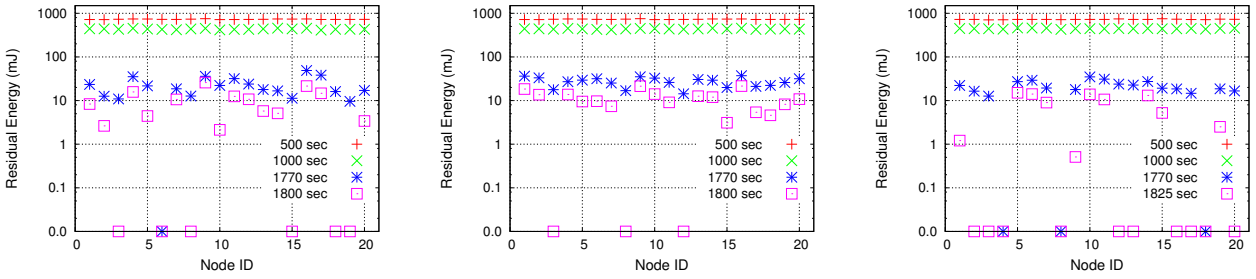

Fig. 1. Snapshots of per-node residual energy in a single sector that resides in the second row and column of the simulated 100m x 100m sensor grid: (a) residual node energy for random aggregator node election; (b) residual node energy for energy-based aggregator node election; and (c) residual energy for connectivity-based aggregator node election. Although the *y axis* is logarithmically scaled, we adjust the scale such that it depicts 0mJ residual energy values.

energy consumption model, each such message transmission and reception costs 56430 and 54000 nJ, respectively. The aggregator fuses the received 120 byte messages, thus the energy cost of computation 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 that resides in the center of the area. This transmission costs 98423 nJ. Hence, the aggregator incurs the highest cost for computation, and for message reception and transmission.

By analyzing our simulation logs, we observe that at the early stages of the network's life, sensor nodes expend roughly 0.5mJ per aggregation phase (per second), while aggregators expend roughly 1.5mJ. In the randomized protocol, nodes become aggregators once every 20 elections (100 seconds), on average. In the connectivity-based approach, nodes have 12-19 one-hop neighbors in their sector and usually only 2-4 highly connected nodes have 18-19 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 remaining energy. In Figure 1, we depict 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 latter 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 20 nodes are completely energy depleted; and b) in the energy-based election, 3 out of the 20 nodes are depleted. These results do 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 connectivity-based election, we observe that at the early stages the energy levels of the three nodes (4, 8 and 18), with the most neighbors (18-19) 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 one node is exhausted in the random and no nodes in the residual-energy-based election.

These results indicate that for a common application scenario such as the one we simulate, random election does not yield highly uneven energy consumption. Hence, it does not substantially reduce the network lifetime. However, we observe that random election is less suitable than the residual-energy-based approach when the network's energy is close to exhaustion. The results also show that connectivity-based approaches result to highly connected nodes to be exhausted faster. Consequently, the network loses substantial connectivity in its early life stages. Moreover, at the early stages of the network's life the network is dense, therefore connectivity is ensured regardless of election criterion, even though it may not be optimal.

IX. Related Work

In this section we compare our work to existing cluster head election protocols. We classify prior work as follows:

Fully Randomized. Wang et al. [20] recently 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 do not need to consider the resource scarcity of WSNs.

Probabilistic Residual-energy-based. LEACH [12] 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 [25] is also a probabilistic 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, which is 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.

Both LEACH and HEED allow an adversarial node to announce itself a cluster head and affect the clustering decisions of honest nodes, thus they are not suitable for our threat model.

Degree-based Protocols. Kuhn et al.'s [13] scheme creates minimum connected dominating sets by computing an asymptotically optimal clustering The proposed protocol sets up periodic sleep/listen schedules within a cluster in a quick and energy-efficient way. The goal is to derive efficient transmission schedules between nodes and cluster heads to avoid collisions and loss of connectivity. With respect to our design space, the main pitfalls of this approach are: a) each node can influence other honest nodes' decisions by announcing itself as cluster head (dominator); and b) it tends to favor aggregators with high degree, forming dense clusters and resulting in quick battery drainage of highly connected nodes.

ACE [7] favors nodes with high degree in electing aggregators and 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 cluster head decides which of its followers would have the greatest number of followers and delegates the cluster head role to it. Since it favors nodes with high degree, it results in uneven load distribution. In addition, from our security perspective, it is not desirable to have the current cluster head deciding on which node will be the next cluster head.

Generic-weight-based. *DCA* [4] builds a dominating set of high weight nodes. The weight may depend on node degree, residual energy and other metrics. Before taking a cluster head election decision, a node waits for his one-hop neighbors with higher weights to decide on whether to become cluster heads or to join existing clusters. Upon receiving one or more cluster head announcements, a node n follows the cluster head 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 cluster head. The 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. Consequently, this algorithm allows an adversary to manipulate the election decision.

ID-based. Baker et al.'s *Linked Cluster Algorithm* [3] 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 ys 1-hop neighborhood. The LCA heuristic was revised in [10] to decrease the number of cluster heads produced in the original LCA. With respect to network lifetime, LCA tends to favor nodes with specific IDs, resulting in those nodes incurring disproportionate energy consumption.

Amis et al. [2] 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. This heuristic minimizes the number of cluster heads and it fairly distributes load among cluster heads.

Both the discussed ID-based schemes allow adversarial nodes to fake their ID, and thereby introduce bias in the election process.

Voting-based. Qin et al. [18] 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. The protocol is completely distributed and does not make any assumptions about sensor location and network topology. 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 self-reported degree and proportional to the self-reported residual energy of node A. As a result, it is manipulable by nodes that report fake weights.

Connectivity-aware and Location-aware Protocols for Routing. PEAS [24] extends system lifetime by keeping only a necessary set of sensors to act as forwarders and putting the rest into sleep mode. Sleeping nodes occasionally and independently wake up to probe the local environment and decide to become forwarders if they do not receive any response.

In ASCENT [6], each node assesses its connectivity and adapts its participation in the multi-hop forwarding topology. This system achieves linear increase in energy savings as a function of the network density and the convergence time required in case of node failures.

In both ASCENT and PEAS, an adversarial node can announce itself as forwarder and cause honest nodes to use them as the cluster's forwarders.

GAF [23] and SPAN [9], exploit node redundancy and location information to form energy efficient network topologies. GAF selects particular nodes in a region 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 region 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 for two hop connectivity. With respect to our security criteria, they are both vulnerable to adversarial nodes that fake their location or persistently announce themselves as forwarders.

X. Conclusion

We design and compare three practical and security enhanced aggregator election protocols for WSNs. The security of these protocols is defined as the *non-manipulability* of the election process by an adversary that aims at the election of nodes controlled by him. Security is achieved by randomizing the election process and preventing any participating node from predicting the election outcome prior to committing to its election input. Our simulation-based evaluation shows that in common application scenarios, randomized selection of aggregator nodes yields satisfactorily balanced energy distribution, when compared to residual-energy- and connectivity-based election protocols.

We conclude that our protocols are capable of ensuring election non-manipulability, while fulfilling several desirable properties with respect to *correctness* and *performance*.

References

[1] M. Acharya, J. Girao, and D. Westhoff. Secure Comparison of Encrypted Data in Wireless Sensor Networks. In *WiOpt*, 2005.

[2] A. D. Amis, R. Prakash, D. Huynh, and T. Vuong. Max-Min D-Cluster Formation in Wireless Ad Hoc Networks. In *INFOCOM*, 2000.

[3] D. J. Baker and A. Ephremides. The Architectural Organization of a Mobile Radio Network via a Distributed Algorithm. In *IEEE ToC*, 1981.

[4] S. Basagni. Distributed Clustering for Ad Hoc Networks. In *I-SPAN*, 1999.

[5] G. T. C. Castelluccia, E. Mykletun. Efficient Aggregation of Encrypted data in Wireless Sensor Networks. In *MOBIQUITOUS*, 2005.

[6] A. Cerpa and D. Estrin. ASCENT: Adaptive Self-Configuring sEnsor Networks Topologies. In *IEEE TMC*, 2004.

[7] H. Chan and A. Perrig. ACE: An emergent algorithm for highly uniform cluster formation. In *EWSN*, 2004.

[8] H. Chan, A. Perrig, and D. Song. Secure Hierarchical In-Network Aggregation in Sensor Networks. In *ACM CCS*, 2006.

[9] 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.

[10] 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.

[11] S. Ganeriwal, R. Kumar, and M. B. Srivastava. Timing-sync Protocol for Sensor Networks. In *ACM SenSys*, 2003.

[12] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. An Application-Specific Protocol Architecture for Wireless Microsensor Networks. In *IEEE TWC*, 2002.

[13] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Initializing Newly Deployed Ad Hoc and Sensor Networks. In *MOBICOM*, 2004.

[14] L. Lamport, R. Shostak, and M. Pease. The Byzantine Generals Problem. In *ACM TPLS*, 1982.

[15] R. C. Merkle. Secure Communications over Insecure Channels. In *Communications of the ACM*, 1978.

[16] A. Perrig, R. Szewczyk, V. Wen, D. E. Culler, and J. D. Tygar. SPINS: Security Protocols for Sensor Networks. In *WINE*, 2001.

[17] B. Przydatek, D. Song, and A. Perrig. SIA: Secure Information Aggregation in Sensor Networks. In *ACM SenSys*, 2003.

[18] M. Qin and R. Zimmermann. An Energy-Efficient Voting-Based Clustering Algorithm for Sensor Networks. In *SNPD*, 2005.

[19] K. Sun, P. Ning, and C. Wang. TinySeRSync: Secure and Resilient Time Synchronization in Wireless Sensor Networks. In *CCS*, 2006.

[20] X. Wang, X. Guo, and X. Yin. NRCE: A New Random Cluster Election Algorithm. In *Journal of Communication and Computer*.

[21] 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.

[22] Z. Xiang, R. Bagrodia, and M. Gerla. GloMoSim: A Library for Parallel Simulation of Large-scale Wireless Networks. In *PADS Workshop*, 1998.

[23] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed Energy Conservation for Ad Hoc Routing. In *MOBICOM*, 2001.

[24] 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.

[25] 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 [15]. 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. Thereafter, this key serves as a shared secret between A and B. An eavesdropper only acquires knowledge of the ciphertexts and the identifier. As he doesn't know to which ciphertext the identifier belongs, his only option is to brute-force 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

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

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 $x \mapsto x^e \mod n$. 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 is, 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.