K.G. Saur Verlag, München, 2009

J. Wilke, E.-O. Blaß, F.C. Freiling, and M. Zitterbart

A Framework for Probabilistic, Authentic Aggregation in Wireless Sensor Networks

Joachim Wilke studied Computer Sciences at Universität Karlsruhe (TH) and received its diploma degree in 2007. Since then, he has been working as a scientific assistant at the Institute of Telematics in Karlsruhe. His main research interests include security and robustness in wireless sensor networks. Currently, he is working on robust, energy-aware concast communication schemes in wireless sensor networks within the BW-FIT project "ZeuS".

Dr. *Erik-Oliver Blass* recently joined Eurecom's Network Security team in Sophia Antipolis, France, as a postdoc. His current work focuses on security and privacy aspects of RFID systems, in particular basic communication protocols between 'tags' and 'readers'. Before joining Eurecom, he received his diploma and doctorate from Institut für Telematik at Universität Karlsruhe (TH). In his previous work, he investigated new security aspects of wireless sensor networks, where strong adversaries are able to compromise resource-limited nodes, and therefore only 'better-than-nothing security' guarantees are possible.

Felix Freiling holds a Diploma and PhD in Computer Science from Technical University of Darmstadt, Germany. He is now a professor for Computer Science at University of Mannheim, Germany. His research interests are in the theory and practice of dependable and secure systems.

Martina Zitterbart is full professor in computer science at Universität Karlsruhe (TH). From 1987 to 1995 she was research assistant in Karlsruhe, receiving her doctoral degree in 1990. From 1991-1992 she was on leave of absence as a Visiting Scientist at the IBM T.J. Watson Research Center, Yorktown-Height, NY. She was Visiting Professor at the University of Magdeburg and the University of Mannheim and full professor at the Technical University of Braunschweig (1995-2001). Her primary research interests are in the areas of multimedia communication systems, mobile and ubiquituous computing, ambient technologies as well as wireless sensor networks. She is member of the IEEE, ACM and the German Gesellschaft für Informatik. In 2002, Martina Zitterbart received the Alcatel SEL research award on technical communication. In 2003, she was General Co-Chair of the ACM SIGCOMM conference which was held in Karlsruhe.

ABSTRACT

Wireless sensor networks suffer from limited resources, in particular finite energy supply. A common way to save energy is reducing radio transmissions by using in-network data aggregation, which is very sensitive to non-cooperative nodes, e.g., nodes that have been compromised by an attacker. Usually, security to such attacks can only be achieved by spending considerably more energy. In this paper, we present the ESAWN framework, a highly customizable protocol for secure in-network data aggregation. The main contribution of ESAWN is providing gracefully degrading security guarantees, in particular dataauthenticity. Instead of providing 'full' authenticity, we only assure an aggregate to be authentic with a given probability. This is done by propagating aggregates on redundant paths, allowing other nodes to check correctness. Hence, a user can trade-off security against energy in a very fine-grained manner. We present both analytical and MICA2-based simulation results, showing the practicality of our approach. For example, with 10% compromised nodes, ESAWN saves, up to 70% energy while degrading authenticity by 5%.

I INTRODUCTION

Data aggregation: A typical task of wireless sensor networks is the measurement of data and its transportation towards a data sink. In-network data aggregation is the most common way to reduce network communication and thus energy-consumption [1]. For this, each node that receives measurements aggregates them using a defined *aggregation function* and forwards the resulting *aggregate* to the sink. This substantially reduces the data-volume to transmit and saves a lot of energy compared to simply forwarding each individual measurement to the sink.

Fig. 1 shows a sensor network used for fire detection. Four sensor nodes a, b, c, and d measure the room temperature and report their measurements as 'node;temperature' towards the sink. On each intermediate hop, there is the possibility of data-fusion by aggregation of incoming data and simple forwarding

10.1515/piko.2009.0021

of a possible fire's location. If the reported temperature exceeds 100ºC, the aggregation node sends '1;*areacode*' and '0' otherwise. In this example, node x receives input from nodes a and b and sends '1;A' to node z. This is interpreted as the detection of a fire in area A. Node z combines the reports of nodes x and y, so an aggregate can be an input value for another aggregation, i.e., the aggregation *cascades*. An *aggregation-tree* represents the hierarchy created by the different aggregation paths in the sensor network.

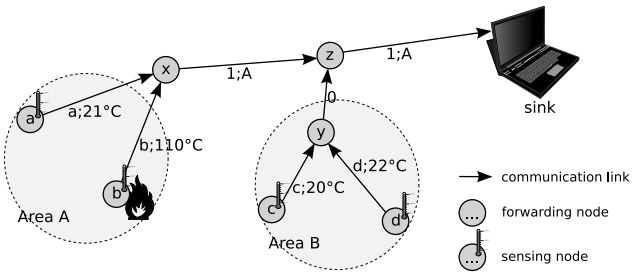


Fig. 1 Example for data aggregation in sensor networks

Compromised nodes: If the sensor data is sensitive or valuable, the aggregation process must be protected by a security mechanism. Since sensor nodes are typically not tamper-proof, physical access to a sensor node allows an attacker to gain not only knowledge of the data stored on this nodes, but also enables him to reprogram the node to achieve his goals [2]. This threat leads to the question of how to perform data aggregation securely, i.e., how to compute correct aggregates even if nodes within the aggregation tree behave maliciously. To illustrate the problem, think of node y in Fig. 1 being compromised by an attacker, i.e., this node behaves maliciously. Node y could report '1;B', even if c reported something different.

Authentic aggregation: There are many security properties of data aggregation. The one which we are interested in is *authenticity*. Authenticity of data aggregation means that all aggregates computed within the aggregation tree reflect the true results of the measurements, i.e., the result is unbiased. In the example sketched in Fig. 1, neither z nor the data sink have the possibility to verify the correctness of the aggregation if y behaves maliciously. Thus, y can influence the aggregation in such a way that false aggregates are computed without being detected. As a result, aggregation provides no authenticity.

A Main Contributions

This article proposes a novel, generic framework for probabilistic, authentic aggregation in wireless sensor networks. Based on ideas, first suggested as *basic* ESAWN [3], [4], we introduce major security improvements, not provided before, e.g., *Non-Repudiation*, which aims at *identifying* misbehaving nodes.

As with the basic ESAWN protocol, our framework focuses on mid-size sensor networks of some thousands of nodes. Our protocol allows *arbitrary* aggregation functions and enables an authenticity-energy *tradeoff*. Furthermore, it improves the *resilience* and adds the security property of *Non-Repudiation*. We propose an abstract *meta protocol* that can be instantiated with different parameters to yield protocols of different strengths and energy costs.

In the framework, the user can specify the probability P that an aggregate finally received at the data sink is authentic. The higher P is set, the higher the resulting energy consumption and the higher the security level provided. Furthermore, per node energy and memory consumption are *independent* to the total number of nodes in the network and scale with $O(1)$.

The ESAWN framework can be instantiated in multiple ways. In this paper, we discuss three increasingly powerful instances called ESAWN-1, ESAWN-2, and ESAWN-NR (see Fig. 2). These three instances differ in the amount of effort, spent to detect forged aggregates by varying datatransport, -encryption and -aggregation. ESAWN-2 adds resilience, i.e., faulty aggregation values can be *masked* (Soundness). ESAWN-NR additionally also provides the property of *Non-Repudiation*.

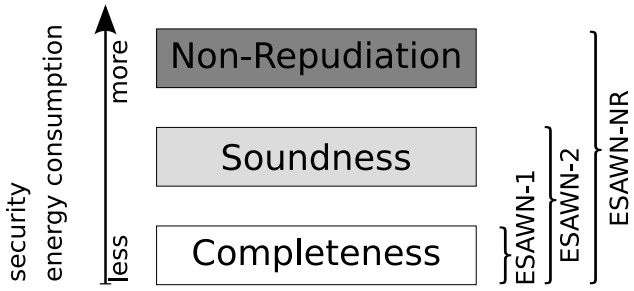


Fig. 2 Different security levels provided by the ESAWN framework

The basic idea of the ESAWN framework is to use a configurable number of redundant aggregation paths along the aggregation tree towards the sink. The aggregated values travel along the redundant paths and are compared by different nodes. The comparison can be used to *detect* and *correct* false aggregation values. Here, the challenge is coping with multiple and cooperating compromised nodes. We assume that the number of misbehaving nodes is given as a fraction β of all nodes n within the network. Just like P, β is also an protocol parameter.

As additional contributions we formalize the concept of authentic aggregation, we provide proof sketches of the correctness of the framework, and we provide *analytical* and *empirical* results that document the powerful energy-security tradeoff of the framework protocols.

B Outline of paper

The remainder of this paper is structured as follows: Section II gives the formal system model. Section III formalizes the problem of authentic aggregation. Section IV describes the protocol framework in detail whereas Section VI evaluates it. A correctness proof is given in Section V. Finally, Section VIII concludes this paper.

II SYSTEM MODEL

The following assumptions are made regarding aggregation trees and functions, key distribution and the attacker model.

A Aggregation Trees and Network Topology

We model a sensor network as an aggregation tree $T = (V, \rightarrow)$ where V is the set of sensor nodes and $\rightarrow \subset V \times V$ is the aggre-

gation relation. If $v \to v'$, we say that v' has an incoming edge (from v) and v has an outgoing edge (to v'). We assume $|V| = n$, and each sensor node in V has an unique identifier.

The aggregation relation satisfies the following properties:

1) There exists a unique node $r \in V$ that has only incoming edges. This node is called root node.
2) Every node $v \in V \setminus \{r\}$ has exactly one outgoing edge $v \to v'$. The node v' is called the *parent* of v and v is called a *source* of v'.

The set of aggregation nodes is A_V. For every aggregation node $v \in A_V$ the number of source nodes is called δ_v. The arithmetic mean of all δ_i of all aggregation nodes is denoted δ. Nodes without incoming edges are called *leaf nodes*. Leaf nodes represent sensors which only measure environmental data while the root node represents the base station or data sink.

As an example, consider Fig. 1, where V is defined as $V = \{a, b, c, d, x, y, z, \text{sink}\}$ and the aggregation relation is defined by the set $\{a \to x, b \to x, c \to y, d \to y, x \to z, y \to z, z \to \text{sink}\}$. Furthermore, $\delta = 2$ and the set of leaf nodes is $\{a, b, c, d\}$.

In practice, aggregation trees are built using standard mechanisms (see for example Intanagonwiwat et al. [5] or Fasolo et al. [1]). We assume, that our *network topology* permits direct communication between nodes being adjacent in the aggregation tree. Further links are *not* required, as all communication is done *multi-hop* along the aggregation tree's edges.

B Aggregation Functions & Traffic Model

For simplicity, we assume a sensor network which periodically measures data. Measurements flow along the tree towards the root. During this process they are aggregated, as soon as v received all incoming data. Different traffic models may have influence, e.g., with regard to the aggregation ratio, but this is out of scope in this work.

We postulate an *aggregation function* F_v that v uses to compute agg_v, the *aggregate* of the data of all of the sources of node v. In general, aggregation can *cascade*, i.e., a node can be both, an aggregation node and, simultaneously, a source node for another aggregation node.

C Ancestors, Descendents and Paths

For any node v, we recursively define for any $i \geq 0$ the i-ancestors of *node* v as follows: Node v is the 0-ancestor of v. If v' is a i-ancestor of v and $v' \to v''$ then v'' is a $(i + 1)$-ancestor of v.

For any node v and any $i \geq 0$, we define the set of i-descendants of v to be the set of all nodes for which v is the i-ancestor. So while ancestors define nodes in the direction towards the root, descendants define nodes in the direction towards leaf nodes. In Fig. 1 for example, the sink is the root node of the aggregation tree, z is the 1-descendant of the sink and x is a 2-descendant of the sink. Conversely, z is a 1-ancestor of x and a 2-ancestor of c for example.

A *path* in T is a sequence of nodes $P = \langle v_1, v_2, \ldots, v_l \rangle$ such that $\forall i \in \{1, \ldots, l-1\} : v_i \to v_{i+1}$.

D Communication

We assume robust communication along the edges of the aggregation tree, e.g., by the use of a *Stop-and-Wait*-mechanism as provided by TinyOS [6]. Higher bit error rates require more retransmissions to compensate packet loss. Here, ESAWN makes no difference to other communication protocols. We focus on efficient communication at the *application* level, instead.

E Key Setup and Encryption

We assume that any two nodes $v, v' \in V$ which need to communicate share a pairwise symmetric key $K_{v,v'}$ for encryption of all data. Such keys must have been established in advance using an appropriate mechanism, e.g., [7], [8].

As we will see later, the ESAWN framework requires that nodes communicate only with a constant part of the entire set of sensors. Every node therefore has to establish and store $O(1)$ keys. A detailed memory analysis is given in Section VI.

All messages sent between nodes are encrypted and authenticated using these pairwise keys. To assert authenticity, *authenticated encryption* [9] is used. This allows the receiver to check if the received data package was transmitted from the assumed sender node.

We assume a fixed aggregation tree built up after deployment. Changes in the aggregation relations, e.g., because of node movement, need an adaption in key distribution and local node knowledge, too. However this is out of scope of this framework and should be addressed by choosing an appropriate key distribution protocol.

F Attacker Model and Security Parameters

A node can either be honest or compromised. An honest node correctly follows the protocol. A *compromised* node can behave arbitrarily. Arbitrary behavior models the result of an attacker that can take over full control of the node in the network. Full control implies knowledge about stored keys and data as well as the ability to reprogram the node.

We assume a *uniformly distributed, static* attacker, i.e., an attacker that may choose to compromise a certain fraction β of nodes in advance, but the attacker cannot later modify his choice and chooses his victim-nodes randomly and uniformly distributed. Moreover, sybil attacks, i.e., node cloning, to increase the fraction of nodes which are under the attacker's control, are impractical, as β is calculated out of the number of *unique* nodes, having own key material.

An adaptive attacker, knowing in advance which nodes to attack for best results, is out of scope in this work. In fact, a protocol for verifying authenticity of aggregates respecting this attacker and all previously defined criteria, e.g., arbitrary aggregation functions, would be hard if not impossible to create in sensor networks.

The choice of β determines the *security parameter* (t, k) of the protocol framework. Parameter (t, k) means that at most k out of t consecutive nodes on any path in the aggregation tree could be compromised. Choosing k the right way is important

for the security-level provided by the framework. Assuming a fraction of compromised nodes β equally distributed in the network, this can be expressed as $k = \beta \cdot t$. Because the attacker compromises randomly, there may be cases where more than k consecutive nodes are compromised. In this case, we cannot detect any fraud, which decreases P. The probability for this and how to come up against this is evaluated later in Section VI-E.

The value t is chosen to be the minimum value needed to assert the security guarantees. To achieve completeness, $t \geq k + 1$ is required, thus $t = k + 1$ is chosen. To achieve soundness, $t \geq 2k + 1$ is required, thus $t = 2k + 1$ is chosen. So k should be at least set to $\lceil \frac{\beta}{1-\beta} \rceil$ in case of $t = k$+1 and to $\lceil \frac{\beta}{1-2\beta} \rceil$ in case of $t = 2k$+1. Completeness and soundness are two of three security properties our framework will provide. See Fig. 2 and Section III for details on this.

Note that a value $t = 0$ means that there is no security built into the protocol, i.e., we do insecure aggregation. Intuitively, the larger t gets, the more security the protocols offer and the higher the energy consumption is.

As we are using multi-hop communication, we assume the attacker not to start any kind of denial-of-service (DoS) attacks in the network, e.g., denial of forwarding messages. Instead we assume the adversary's incentive to be unnoticed injection of faulty aggregates or similar attacks against the network. The absence of messages somewhere in the aggregation tree could be a mean to detect a DoS event. However, because of the complexity of this topic, we keep this for future research.

We assume the sink to be honest since authentic aggregation would be impossible otherwise. This is in accordance with other literature [10]-[12]. Under certain (extensive) assumptions and constraints like stochastic correlation of multiple measurements [13] it may be possible to detect manipulations of nodes' measurements. Our framework focuses on authentic *aggregation*, so we assume that nodes are honest with respect to their measurements. However, ESAWN can be extended with RANBAR/RANSAC mechanism, if measurement verification is needed.

In other work [10]-[12], it is assumed, that only leaf nodes actually measure data. Inner (i.e., non-leaf) nodes only compute the aggregation of their sources. With this framework however, every node can contribute its measurements to the aggregation, as measurements and aggregate calculations on the node are strictly separated by the protocol.

III AUTHENTIC AGGREGATION

An honest aggregator node v *accepts* an aggregate $agg_{v'}$ of one of its source nodes v', if v uses $agg_{v'}$ in its own aggregation process, i.e., v computes its aggregate $agg_v = F_v(\ldots, agg_{v'}, \ldots)$ and forwards it to its parent in the aggregation tree. If v is compromised, then v may compute an aggregate that is *forged*, i.e., which is not equal to the result of applying F_v to the data of all of v's source nodes. An aggregate is called the *correct* aggregate if it is not forged, i.e., if it is equal to the result of applying F_v to the data of all of v's source nodes.

We define the following properties for an algorithm that solves *authentic aggregation:*

- Completeness: No honest node accepts a forged aggregate.
- Soundness: An honest node accepts every aggregate, i.e., the node can *mask* forgeries of its source nodes when computing its own aggregate.
- Non-Repudiation (NR): Every honest node v can prove to a third party that it accepted a correct aggregate if and only if it accepted a correct aggregate. Here, a third party can be any node with which v shares a secret key.

In ESAWN, to satisfy Completeness we employ a mechanism to *detect* forged aggregates. To satisfy Soundness *and* Completeness, we introduce a mechanism being able to *detect* and *correct* forged aggregates. To realize Non-Repudiation, we use logging of crypted communication.

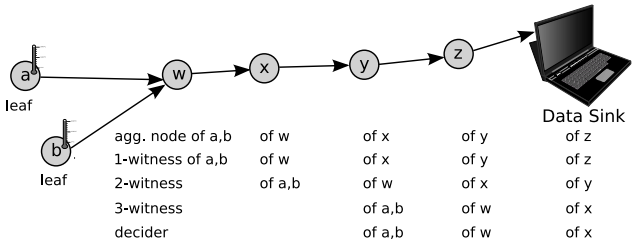


Fig. 3 Naming example, t = 3

IV ESAWN FRAMEWORK

Every ESAWN protocol instance guarantees at least Completeness of authentic aggregation under the assumption that $t \geq (k+1)$. In case $t \geq (2k+1)$ our framework also guarantees Soundness and is able to realize Non Repudiation (see Fig. 2).

A Overview

The idea of the ESAWN protocol framework is to propagate aggregates on redundant paths, thereby allowing other nodes to check the correct aggregation. The ESAWN protocol framework is based on the concept of witness and decider nodes. For all of its j-descendents, $0 \leq j < t$, node v is a *witness node*. Note that the 0-witness of v is v itself. If a 0-witness v has no sources, then v is a leaf node. Otherwise, v is an aggregation node for its sources. For any node v, we define the *decider node* of v to be the t-ancestor of v. An example is shown in Fig. 3.

Roughly speaking, witness nodes of v redundantly compute the same aggregate as v does. Then, the decider node of v uses this information of the witnesses of v to determine the correct aggregate or to detect at least a forged aggregate as shown below.

B Protocol parameters

The ESAWN framework provides two user configurable parameters, P and β, and three types or properties (Completeness, Soundness, Non-Repudiation) to adjust the energysecurity tradeoff. From these parameters, concrete values for (t, k) are chosen to assert the demanded level of authenticity P with respect to the fraction of β compromised nodes, as described in Section II-F. An additional protocol parameter p specifies the probability of node verification. The higher p is, the higher is the probability that an aggregation node is checked and the higher is the energy consumption of the protocol.

C ESAWN Protocol Framework

The framework follows these simple rules:

1) A measuring node v regularly measures its environment and sends its measurement to v's t witnesses and v's decider.
2) An aggregating node v waits for values from all of its sources, computes the aggregation function F_v on these values and sends the resulting aggregate to v's decider.
3) A witnesses of a node v waits for values from all of the sources of v, computes the aggregation function F_v on these values and sends the resulting aggregate to the decider of v.
4) The decider v' of node v waits for messages from all witnesses of v. It then performs a majority vote on the received values. If there are at least two different values then stopping the protocol for this aggregate achieves *Completeness* (this results in ESAWN-1). If there is a majority of values, then the majority value is accepted by v' to be the correct value of v to achieve *Soundness* (this results in ESAWN-2).
5) Witness calculations and the majority vote at the decider node are done *synchronised* with probability p. Synchronisation can be implemented using the a shared seed or internal state for a pseudorandom number generator [14].
6) In case there were differing values at a decider of v, it can start an identification protocol to identify dishonest nodes (ESAWN-NR). This consists of disclosing the keys the decider of v used to encrypt its communication with v and v's witnesses. Using these keys, all witness nodes can verify the report by decrypting the commitment they stored before. See Section IV-D for details.

Fig. 4 shows the algorithm in pseudocode presented as local algorithms of sensor nodes, segmented according to the roles that the node might play. For simplicity, probabilistic verification is omitted in the pseudocode, i.e., p = 100% is assumed.

Remember that any node v can take measurement, do aggregation or *both* at a time. In the latter case, it must follow *both* protocol directions, for a measuring *and* for an aggregating node.

Further, all communication is done multi-hop using the shortest paths along the aggregation tree *without* skipping intermediate nodes, other communication links are *not* assumed to be available. While the shortest path is the intuitive alternative for communicating, it is *not* applicable in conjunction with Non-Repudiation. Here, all communication is rerouted via the aggregation node v to enable all witnesses of v and v itself to save their commitment (the sent data). Refer to Section IV-D for details.

As a node communicates with nodes up to t levels upand downwards in the aggregation tree, nodes need knowledge about the tree structure of this constant fraction of the whole tree. All communication is performed using authentic encryption (recall Sect. II-E).

D Identification protocol

Using the framework's instance ESAWN-NR, the decider of a node v can determine compromised nodes that sent faulty aggregates as these aggregates form a minority of at most k out of t values. Because of $t \geq 2k$+1 and the authentic encryption of values, the decider can assign each faulty aggregate to a compromised node.

After that, all nodes that took part in this aggregation (v's witnesses and v itself) get informed about the existence of compromised nodes. To prove the report, the decider of v reveals the communication keys it shares with v and v's witnesses.

```
1  Code for aggregating node v :
2      // v in the role as decider of v':
3      For all t-descendants v' of v do
4          Wait for data from v' and all j-witnesses of
           v' (1 ≤ j < t).
5          Perform majority vote
6          if mismatch detected then
7              if only achieve Completeness then
8                  stop protocol
9              if achieve Non Repudiation then
10                 start identification protocol
11         // to achieve Completeness and Soundness
12         Accept majority
13     // v in the role as j-witness of v' (0 < j < t):
14     For all j from t − 1 downto 1 do
15         For all j-descendants v' of v do
16             Wait for data from all sources of v'.
17             Compute result r of aggregation function
               F_v' for v'
18             Send r to decider of v'
19     // v in the role as aggregator, i.e., 0-descendant of
       v:
20     Wait for data from all sources of v.
21     Compute aggregation function F_v for v
22     Send result to decider of v
23 Code for measuring node v :
24     // v in the role as leaf node ("0-witness of v"):
25     Wait for next timing period to begin
26     Measure value
27     Send value to all j-witnesses of v (1 ≤ j < t)
       and decider of v
```

Fig. 4 ESAWN Pseudocode

These nodes can now verify the report by using the keys provided to decrypt the previously logged data. This data contains the encrypted values that passed the node during communication. Remember, that all communication is routed via v. As a result, all nodes logged the needed information to verify a report.

Because an attacker could provoke key revelation to get the keys, it must be noted, that revealed keys can no longer be used for securing communication. Instead new keys must be distributed using an appropriate mechanism, see Section II-E. The implied overhead is expected to be low, because the detection of compromised node is not the normal case.

E Examples

In the following, all three ESAWN instances are illustrated by focusing on a sub-path of an aggregation path and describing the actions taken there by the protocol. To keep illustrations simple, we assume δ = 1 and k = 1.

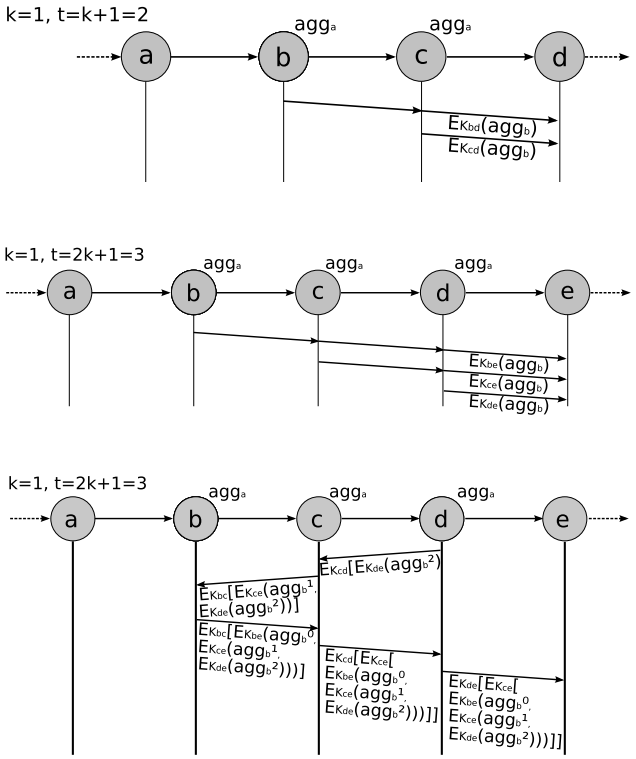


Fig. 5 Example message flows for all three framework instances

1) *ESAWN-1:* Fig. 5(a) shows a part of an aggregation path currently processing the aggregation at node b using ESAWN-1. Assume that nodes b and c have all the required input data, in this case this is agg_a, to carry out the aggregation and calculate $agg_b = F(agg_a)$.

Then, node b sends its aggregate to its decider d. Node c is the 1-witness of b. It calculates agg_b for itself and sends it to b's decider, too. Both messages are encrypted, using the pairwise symmetric key between the communicating nodes (b, d) and (c, d) respectively. If the aggregates do not match, node d will stop the aggregation (alternatively it could propagate an alarm message towards the sink node to inform it about the fraud).

2) *ESAWN-2:* Fig. 5(b) shows a part of an aggregation path currently processing the aggregation at node b using ESAWN-2. Assume again that nodes b, c, and d have all required input data, in this case this is agg_a, to carry out the aggregation and calculate $agg_b = F(agg_a)$.

Then, node b and both witness nodes, c and d, send their calculated aggregate to node e, the decider node for this step. agg_b^0 represents the value calculated by node b. agg_b^1 and agg_b^0 these of nodes b and c. Node e does a majority-vote on the received values agg_b^0, agg_b^1 and agg_b^2. If at most k values (in this example one value) are wrong here, the majority-vote will find the correct value for the aggregate and compensate this fraud. As with ESAWN-1, all messages are encrypted, using the pairwise symmetric key between the communicating nodes (b, e), (c, e) and (d, e) but communicated multi-hop.

3) *ESAWN-NR:* Fig. 5(c) shows a part of an aggregation path currently processing the aggregation at node b using ESAWN-NR. Regarding to the aggregation calculations, the majority-vote, and the task assignment, this is similar to the previous example using ESAWN-2.

What differs here, is the communication order between the nodes. This is, because ESAWN-NR takes care about every witness node and the aggregation node itself can 'see' what the other nodes send to the decider node if only in an encrypted version. This commitment is used to prevent repudiation of sent data. This is the reason for node d to not directly send its aggregation result to node e but making a detour via node b. The same holds for node c.

The encryption done here is somewhat more complex than with ESAWN-2. This is required to make the forgery of values impossible, once sent from the originator (see Fig. 5(c)).

Imagine c being compromised, i.e., it is $agg_b^0 = agg_b^2 \neq agg_b^1$. Node e detects agg_b^1 differing from the other values and sends an alarm message to nodes b and d including its communication keys (b, e) and (c, e). Node b, that stored $E_{K_{be}}(agg_b^0, E_{K_{ce}}(agg_b^1, E_{K_{de}}(agg_b^2)))$ before, decrypts this package and sees that $agg_b^1 \neq agg_b^0$. Node d, that stored $E_{K_{ce}}(E_{K_{be}}(agg_b^0, E_{K_{ce}}(agg_b^1, E_{K_{de}}(agg_b^2))))$ decrypts this package and sees that $agg_b^1 \neq agg_b^2$. Both are now convinced that c is compromised. Together with e, they inform all their ancestors and the sink of this fraud.

V CORRECTNESS

We now give the main lines of the correctness proof of the ESAWN protocol instances. Before doing this, observe that for $t = 0$, the scheme reduces to *non-authentic* aggregation. To see this, note that if $t = 0$ and $v' \rightarrow v$, then v is the decider of v'. Furthermore, if $t = 0$ then no node has j-witnesses with $j \geq 1$. Therefore, a measuring node sends its measured value only to its decider and an aggregation node aggregates its values and sends it to its decider. The decider does nothing apart from accepting this value in its own aggregation.

So, now consider the case where $t > 0$. We prove this general case by induction over the *level* of an aggregate as it flows along the tree towards the root. The *level* of an aggregate at node v is the longest path from v to a leaf node. For example, the level of a leaf node is 0 and the level of v which has only leaf nodes as sources is 1, etc. The following theorem states that ESAWN satisfies *Completeness*.

Theorem 5.1: For all levels $l \geq 0$ holds: No honest node v at level l accepts a forged value.

To prove Theorem 5.1, we make the following useful observations.

Lemma 5.2: The j-witness of v is a $(j + 1)$-witness of a j-ancestor of v.

To prove the base case of Theorem 5.1, consider node v at level $l < t$. Now two cases can occur: (1) v is a leaf node, or (2) v is not a leaf node. Case 1 is trivial, because a leaf node never accepts anything. In case 2 and because $l < t$, node v cannot be a decider for any node. But v is a witness for other nodes.

In an additional induction over all nodes of which v is witness, we prove the following statement: v in its role as j-witness of v'' computes the correct value of v''. As a base, we consider the largest j such that v is j-witness of v'. In this case, all sources of

v' must be leaves. From the algorithm, every leaf sends its value to every witness, so v receives values from all sources of v'. Since leaves are honest by definition, v computes the correct value of v'. In the induction step consider node v'' of which v is $(j-1)$-witness and assume that v has computed the correct value in its role as j-witness of v'. Using Lemma 5.2, we can show that if v computes the correct value of v', then v also computes the correct value of v''.

In the inductive step of Theorem 5.1, we prove the theorem for all cases where $l \geq t$, assuming that the theorem is true for all levels $l' < l$. If v accepts a value r, then v does this in its role as decider of some node v'. If there is a mismatch in values and $t < 2k + 1$ the algorithm stops. In this case, no further node in higher levels accepts any value and the proof is completed. In case $t \geq 2k + 1$, r was the majority value of the values sent by v' and all witnesses of v'. Since $t \geq 2k+1$ the majority value can only be from honest nodes. From the induction hypothesis, these values were correct and hence v accepts a correct value of v'. This completes the proof of Theorem 5.1.

It is much easier to see that ESAWN satisfies Soundness. The central idea of the proof is the following lemma, which follows directly from the attacker assumption and the use of redundant paths in the ESAWN framework.

Lemma 5.3: If $t \geq 2k + 1$, then every honest decider eventually collects k messages.

The correctness of the value-commitment protocol is at the heart of proving that ESAWN also satisfies Non-Repudiation. The central technical step to follow Non-Repudiation from this is the following lemma.

Lemma 5.4: Every witness of v commits itself towards all other witnesses of v on the value it sends to the decider.

VI EVALUATION

For evaluation purposes, we compare the ESAWN protocols to two simple aggregation protocols:

Non-Authentic Aggregation (NAA): NAA uses standard aggregation as with *directed diffusion* [5] to achieve an efficient but non-authentic data transport. As a protocol for authentic aggregation cannot require less energy than a protocol providing non-authentic aggregation, comparing this framework with NAA as a lower bound, helps to determine the additional expenses for authenticity.

Authentic Non-Aggregation (ANA): ANA uses simple, but secure packet forwarding towards the sink for data transport. To provide security, each leaf node authentically encrypts its measurement using a pairwise secret key shared with the sink. After receiving all data, the sink computes the aggregate itself. As there is no in-network processing, ANA provides traditional authenticity.

A Upper Bounds for Energy Consumption

1) *NAA:* NAA uses no encryption or decryption, so energy costs only arise due to communication. During protocol execution, each node transfers exactly one message, i.e., the calculated aggregate. In total, (n – 1) messages are transported. So, per node energy consumption scales with $O(1)$.

2) *ANA:* ANA uses no aggregation, so each leaf node (δ^h in total) sends its measurement. All other nodes just forward these messages along the aggregation tree towards the sink. The number of hops is the height h of the tree. Because $h = \log_\delta(1 + (\delta - 1)n) - 1$, this means a total of

$$f(n,\delta) := h \cdot \delta^h = (\log_\delta(1 + (\delta - 1)n) - 1) \cdot \frac{n(\delta - 1) + 1}{\delta}$$

messages to be transported. Because of

$$\lim_{n\to\infty} \frac{f(n,\delta)}{n \log_\delta n} = \frac{\delta - 1}{\delta} < \infty$$

per node energy consumption scales with $O(\log n)$.

3) *ESAWN-1:* Every node sends its aggregate encrypted using $(k+1)$ different keys to its parent. This results in $(k+1)$ messages. The parent node forwards k out of this $(k + 1)$ messages, and so on. Summarized, each node sends

$$(k+1) + \sum_{i=1}^{k} \delta^i (k - i + 1)$$

messages. With respect to the total number of nodes n, this scales with $O(1)$.

Second, each node encrypts its aggregate $(k+1)$-many times and decrypts data from all his descendants up to $(k+1)$ levels below, i.e.,

$$\sum_{h=1}^{k+1} \delta^h = \frac{\delta^{k+2} - \delta}{\delta - 1}$$

This scales with $O(1)$ with respect to n, too. So, ESAWN-1's total energy consumption scales with $O(1)$.

4) *ESAWN-2:* An aggregation node x can be i-witness, $i \in \{0 \ldots 2k\}$, for different nodes. Depending on i, x has to send a different number of messages. For x's own aggregation ($i = 0$), x sends one message to its decider. As 1-witness, it sends one message for each of its δ children and forwards one message for each of them. For the i-witness, this sums up to δ^i own messages and $\sum_{j=1}^{i} \delta^j$ forwarded messages. For every $i \in \{0 \ldots 2k\}$, the total number of messages adds up for x to

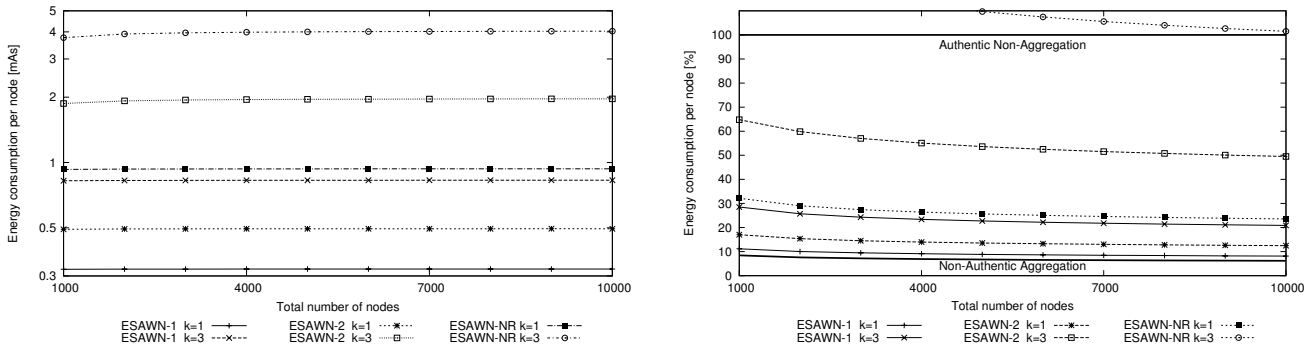
$$\sum_{i=0}^{2k} (\delta^i + \sum_{j=1}^{i} \delta^j)$$

Second, each node encrypts its own aggregate and the aggregates calculated as a witness. This sums up to

$$\sum_{j=0}^{2k} \delta^j = \frac{\delta^{2k+1} - 1}{\delta - 1}$$

encryptions. As a decider, each node needs to decrypt $(2k+1)$ aggregates. As a node is decider for each descendant $(2k+1)$ levels below, this results in $\delta^{2k+1} \cdot (2k + 1)$ decryptions. So, ESAWN-2's total energy consumption also scales with $O(1)$.

5) *ESAWN-NR:* As with ESAWN-2, every node is witness for all its descendants up to $2k$ levels below. In contrast to ESAWN-2, data is not only sent in direction towards the sink node. This increases the total number of messages sent.



(a) Total energy consumption *(b) Relative energy consumption*

Fig. 6 Per node energy consumption of one protocol run, $\delta = 2$

Again, we look at aggregation node x. For x's own aggregation it sends one message containing the aggregate and forwards $2k$ messages, x received from its witnesses. As the first decider, it sends one message to each of the δ children to verify and forwards the $\delta \cdot (2k - 1)$ messages from the other verifiers to them. Thus, as the i-witness x sends $\delta^i(2k + 1 - i)$ messages. In addition, for each aggregation node to verify, x sends $(2k + 1)$ messages towards the decider.

Summarized for every $i \in \{0 \ldots 2k\}$, this results in up to

$$\sum_{i=0}^{2k} \delta^i \cdot (2k+1) + \sum_{i=1}^{2k} (\delta^i \cdot (2k+1-i))$$

messages sent by x.

Second, each node does encryptions like with ESAWN-2, but in addition encrypts all messages sent towards the decider node, to enable Non-Repudiation. So, the total number of encryptions here is

$$\sum_{j=0}^{2k} \sum_{i=0}^{2k-j} \delta^i \cdot (2k+2)$$

The same applies for each decider role, where $(2k+1)$ more decryptions are needed. This sums up to a total of

$$\sum_{i=1}^{2k} \delta^i \cdot (4k+3)$$

decryptions. ESAWN-NR's total energy consumptions also scales with $O(1)$.

6) *Summary:* All protocol instances of our framework scale efficiently with $O(1)$ regarding network size. It must be noted, that increasing k causes an exponential increase of message complexity. However, even large fractions of compromised nodes (e.g., β = 20%), do need only small numbers of k (e.g., k = 3).

B Simulation Results of Energy Consumption

The ESAWN framework was implemented in GloMoSim [15], a discrete event simulation environment, for evaluation and in TinyOS [6] for demonstration purposes [16]. In GloMoSim, so called *runs* were simulated. During a run, each leaf measures a single data item and sends it towards the sink – using the aggregation tree, possibly computing aggregates, and so forth. For aggregation, each node waits for receiving data items from all child nodes in the tree. As we assume reliable data transport, the algorithm finally terminates.

The energy consumption of such a run was calculated out of the number of messages sent and the number of encryption and decryption operations. We also took different, exemplified aggregation functions F_v into account: a single integer addition as a 'cheap' and an RC5-encryption as an 'expensive' aggregation function. However, the impact on the total energy consumption was negligible compared to the energy consumption of message transfer and the large amount of required en- and decryptions.

The simulation is based on the energy consumption and parameters of the MICA2/TinyOS sensor platform [6], [17], [18]. Network sizes of several thousands of nodes, as they are to be expected in the future, are evaluated. With smaller networks, the overhead of authentic aggregation reduces efficiency namable. For each simulation run, the network size n, the degree of aggregation δ and the protocol specific parameters k and p were varied. Table I sums up all important protocol parameters used.

1) *Per node energy consumption:* Fig. 6(a) shows the energy consumption per node of a run with all ESAWN in stances. A complete run is the measurement of environmental data at all leaf nodes and its transportation to the sink node. The shown simulation results use δ = 2 when building an aggregation tree.

Comparing the graphs, which show the energy consumption for ESAWN-1 with k = 1 and k = 3, you can see that a larger k leads to a higher energy consumption. A larger k means using more witness nodes, which results in a higher degree of security but also in a higher number of messages and thus energy consumption. This influence on the energy consumption is similar with ESAWN-2 and ESAWN-NR.

Furthermore, ESAWN-NR consumes more energy than ESAWN-2, and ESAWN-2 consumes more than ESAWN-1 using the same protocol parameters. This is a result of ESAWN-2 requiring twice as much witness nodes as ESAWN-1 for the same k. ESAWN-NR, in contrast, uses the same number of witness nodes as ESAWN-2 does. The higher energy consumption of ESAWN-NR is due to the modification in the protocol's communication 'detours'.

Aside of this unavoidable extra costs for a higher security level, all ESAWN instances scale well: per node energy consumption is bound to $O(1)$ with respect to the total number of nodes.

2) *Energy savings using authentic aggregation:* Using one of the ESAWN instances yields to energy savings compared to Non-Aggregated Authentic communication. The following simulation results show energy-savings possible.

Fig. 6(b) compares all three ESAWN instances using k = 1 and k = 3 to NAA (lower bound) and ANA (upper bound). The amount of energy that can be saved varies with the protocol and its parameters.

Table I Simulation Parameters

simulation environment	GloMoSim [15]
node placement	uniformly distributed
aggregation tree	following n and δ, see Section II-A
message length	56 bytes [18]
measurement size	32 bit
payload	29 bytes [18]
data rate	38,4 kBit/s [18]
radio energy	16 mA [18]
CPU energy	5 mA [19], [17]
energy per message	245 µAs [17]
cipher	RC5 [9]
energy per en-/decryption	1,3 µAs [19]
energy per aggregation	neglected, see Section VI

It is quite interesting that is also possible to choose configurations that are inefficient, regarding energy consumption. This is the case, when the energy consumption exceeds the one of ANA. An example is ESAWN-NR using k = 3 here. In this case it is more energy-efficient to use non-aggregating communication if strict authenticity is needed or to reduce k, resulting in weaker level of authenticity.

C Upper Bounds for Memory consumption

As with energy, memory resources are very limited with current sensor nodes. However, the framework's memory consumption also scales well regarding the total number of nodes in the network.

1) *NAA:* Each aggregation nodes has to manage δ incoming values and one outgoing aggregation value, which sums up to δ + 1 values in memory. This scales with $O(1)$.

2) *ANA:* Each aggregation node only forwards the incoming value, i.e., at one time, there is only one value to manage – scales with $O(1)$.

3) *ESAWN-1:* Each aggregation node v has to manage

$$\sum_{h=1}^{k+1} \delta^h = \frac{\delta^{k+2} - \delta}{\delta - 1}$$

incoming values, i.e., values from all nodes (k + 1)-levels below v in the aggregation tree. These are the nodes that v is witness for. This scales with $O(1)$.

4) *ESAWN-2:* Each node has to manage $(2\,k + 1) \cdot \delta^{2k+1}$ incoming values, i.e., (2k+1) values for all δ^{2k+1} nodes, this node is decider for. This scales with $O(1)$.

5) *ESAWN-NR:* Memory consumption regarding incoming values is identically to ESAWN-2. Regarding Non-Repudiation, each node has to store (2k+1) encrypted values for each aggregation it is aggregation node or witness for. As these are up to $\sum_{h=1}^{2k} \delta^h$ aggregations, memory consumption increases with $O(1)$:

$$(2k+1) \cdot \sum_{h=1}^{2k} \delta^h = \frac{\left(\delta^{2\,k+1} - 1\right)(2\,k+1)}{\delta - 1}$$

D Simulation Results of Memory consumption

In our implementation, memory consumption was examined from different perspectives: average Memory consumption *P*er *N*ode (*MPN*), average Memory consumption *P*er *A*ggregation node only (*MPA*), and the *maximum* of memory consumption (*MAX*) of all nodes. As leaf nodes typically require less memory for the protocols compared to aggregation nodes near the sink, this allows a detailed analysis of memory consumption.

Looking at Fig. 7(a), you can see the memory consumption in all three cases using k = 1. Memory consumption is quite low, as there are only a few nodes to be verified. Compared to this, Fig. 7(b) shows a significant higher memory consumption with k = 3. There is an exponential increase when using higher values for k. However, memory consumption does not increase with the increasing total number of nodes in the network n.

E Probability of Authentic Aggregation P

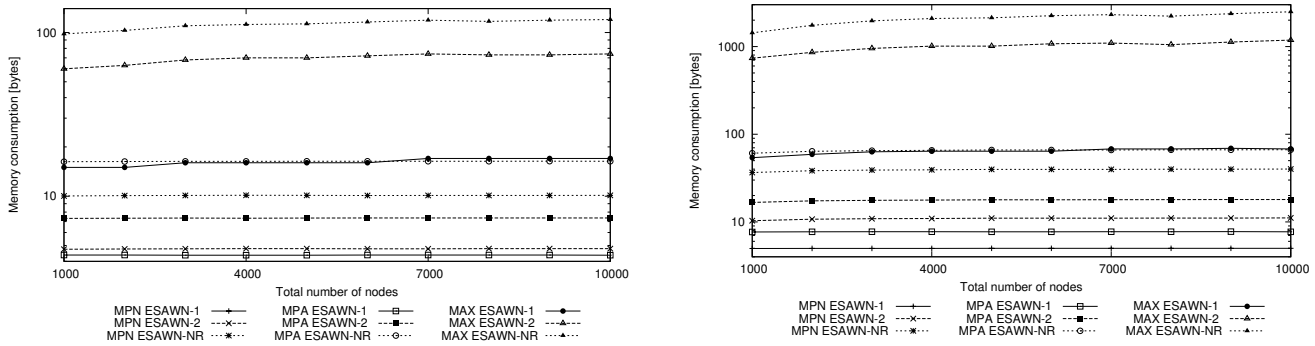
Depending on the ESAWN protocol instance, the network parameters n and β, and the chosen protocol parameters t and p, there is a different probability P for the 'final' aggregate received at the sink being authentic.

For an authentic, 'final' aggregate, every aggregation node has to be honest or, if it is compromised, it must be verified: P = (1 – β + $\beta \cdot p$)$^{|A_V|}$. Here, A_V is the subset of all aggregation nodes A, which is verified during this run, i.e., $\mathcal{A}_V = \sum_{i=t}^{h-1} \delta^i$. Note that $A_V \subset A$, as nodes near the sink are only witnesses and deciders, but do not need to compute their own aggregates, which is done by the sink.

Table II presents simulation results with n = 1000, δ = 2, and varying k, p. With β = 10%, k = 2 and p = 100%, ESAWN-1 asserts an authenticity of around 60% for the sink's aggregate. If the security demand is higher, k needs to be increased. With lower fractions of compromised nodes, e.g., β = 1%, p < 100% can reduce the energy consumption while still providing a significant level of authenticity. As you can see from Table II and Fig. 6(b), even with a high security demand P = 95:3%, a high fraction of compromised nodes β = 10%, and a network of n = 10,000 nodes, more than 30% energy can be saved compared to Authentic Non-Aggregation.

F Summary

We showed using analytical and simulative evaluation, that our framework scales well regarding network size, both in terms of memory consumption and communication complexity. Finally,



(a) k = 1 *(b) k = 3*

Fig. 7 Per node memory consumption

we discussed how n, t and p influence the probability of authentic aggregation $\mathcal{P}$ and highlighted some exemplary configurations. The results show that, with this framework, it is possible to trade off energy consumption and authenticity in a fine grained manner.

VII RELATED WORK

There are various of aggregation techniques available [1]. Most, if not all target the reduction of data volume and thus minmize communication related energy consumption. However, securing aggregation with regard to authenticity is not trivial, especially in wireless sensor networks:

One approach to provide authentic data aggregation is to transport data in encrypted form only and to use privacy homomorphisms to compute functions on encrypted data [20]. While this is an elegant approach, it is only feasible with certain, linear, aggregation functions like addition and multiplication, but not with conditional statements or comparisons within the aggregation function. In addition, privacy homomorphisms are vulnerable to known-plaintext, chosen-plaintext and replay attacks [21]. Finally, nodes aggregating data cannot contribute own measurements into the aggregation.

Table II Possibility of authentic aggregation, using n = 1000 and δ = 2

β	k	p	$\mathcal{P}$ (ESAWN-1)	$\mathcal{P}$ (ESAWN-2/-NR)
1%	1	50%	5.0%	5.2%
1%	1	90%	52.4%	50.1%
1%	1	100%	94.6%	89.5%
1%	2	100%	99.9%	99.7%
10%	1	100%	0.4%	0.1%
10%	2	100%	60.6%	13.7%
10%	3	100%	95.3%	60.1%

SIA [11] and its extension [22] use a statistical verification process by asking a randomly chosen set of nodes for the data they sent. Depending on the aggregation function it is now possible to make statements whether the aggregate is correct or not. While being rather efficient, there is no general mechanism in SIA to deal with arbitrary aggregation functions (up to now only average or median have been investigated). Similarly, SDAP [23] tries to detect outliers, i.e., values differing from a mean. SDAP is also restricted to average and median aggregation functions. Moreover, this approach additionally requires more than one node monitoring the *same phenomenon*, which might be unrealistic in many scenarios. This also applies to *interleaved hop-by-hop authentication* [12], where a majority-vote based protocol for filtering maliciously injected data is presented.

SAWN [10] (Secure Aggregation for Wireless Sensor Networks) uses hash-chains for authenticity, i.e., every sensor node shares a secret with the sink. This secret is the basis for construction of a hash-chain. Every measurement uses the current element of its hash-chain to calculate a commitment that is sent towards the sink. After receiving all commitments, the sink broadcasts the previous elements of all hash-chains into the network. This enables all nodes to verify received commitments and provides security against up to one compromised node. Apart from its low resilience, another drawback of SAWN is that nodes can verify the received aggregate only in a *delayed* manner, i.e., the sink has to collect all commitments first and then re-broadcasts.

Srikanth and Toueg [24] study the problem of authenticated broadcasts in general distributed message passing systems and proposed a simple broadcast protocol based on redundant message paths. This was done, however, using all-to-all communication and without focus on data aggregation.

Proof Sketches [25] is an approach to provide a verification mechanism using cryptographic signatures and *Flajolet-Martin* sketches. However, this needs a central public key infrastructure and complex cryptographic operations, which are both not common in wireless sensor networks.

In ESAWN [3] (Extended SAWN), nodes achieve authentication by using redundant message paths which are recursively defined through the aggregation tree. ESAWN allows use of *arbitrary aggregation functions* but ensures only the most basic authentication: If the sink receives a result, then it is authentic. An attacker can therefore easily perform a denial-of-service attack on the protocol. To save energy, ESAWN proposes to perform authenticity checks only with a certain probability p. Such authenticity differs considerably from conventional definitions of authenticity assuming classical *strict* authenticity, i.e., authenticity of each piece of data with probability 100%. Thus ESAWN exhibits a tradeoff between authenticity and energy. Being an extended abstract, the paper [3] lacks proper formalization and correctness proofs. This was improved later [4], but for the basic ESAWN protocol only.

VIII CONCLUSION

Saving energy contradicts authenticity verification. In this paper, we presented the ESAWN protocol framework that provides a user customizable security vs. energy trade-off for aggregation in sensor networks. We presented three protocol instances, each offering different levels of authenticity resulting in different energy costs.

Furthermore, the ESAWN framework allows to probabilistically relax authenticity, resulting in clearly weaker security, but also less energy consumption. Depending on the user's requirements for security and energy-saving, he can tailor all three protocol instances and setup their parameters. Simulations indicate that large energy savings are possible. For example, with 1% compromised nodes, ESAWN saves between up to 50% energy by degrading authenticity by 5-10% while still providing Non-Repudiation. Without, even 90% of energy savings are possible.

REFERENCES

[1] E. Fasolo; M. Rosso; J. Widmer; and M. Zorzi: "In-network aggregation techniques for wireless sensor networks: A survey," IEEE Wireless Communications, vol. 14, no. 2, pp. 70-87, 2007, ISSN 15361284.

[2] A. Becher; Z. Benenson; and M. Dornseif: "Tampering with motes: Realworld physical attacks on wireless sensor networks." In: Proceedings of International Conference on Security in Pervasive Computing. York, UK: Springer, Apr 2006, pp. 104-118.

[3] E.-O. Blaß; J. Wilke; and M. Zitterbart: "A security-energy trade-off for authentic aggregation in sensor networks (extended abstract)." In: IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), IEEE Computer Society. Washington D.C., USA: IEEE Computer Society, Sep 2006, pp. 135-137, ISBN 1-42440-732-X.

[4] E.-O. Blaß; J. Wilke; and M. Zitterbart: "Relaxed Authenticity for Data Aggregation in Wireless Sensor Networks." Istanbul, Turkey: 4th International Conference on Security and Privacy in Communication Networks (SecureComm 2008), Sep. 2008, ISBN 978-1-60558-241-2.

[5] C. Intanagonwiwat; R. Govindan; D. Estrin; J. Heidemann; and F. Silva: "Directed diffusion for wireless sensor networking." In: ACM Transactions on Networking, no. 11, Jan 2003, pp. 2-16.

[6] UC Berkeley: "Tinyos," 2005, http://www.tinyos.net/.

[7] L. Eschenauer and V. Gligor: "A key management scheme for distributed sensor networks." In: Proceedings of ACM Computer and Communications Security, Washington D.C. USA, Nov 2002, pp. 41-47.

[8] M. Zitterbart and E.-O. Blaß: "An Efficient Key Establishment Scheme for Secure Aggregating Sensor Networks." In: ACM Symposium on Information, Computer and Communications Security, Taipei, Taiwan, Mar. 2006, pp. 303-310, ISBN 1-59593-272-0.

[9] H.C. van Tilborg: Encyclopedia of Cryptography and Security. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.

[10] L. Hu and D. Evans: "Secure aggregation for wireless networks." In: Proceedings of IEEE Symposium on Applications and the Internet Workshops (SAINT). Orlando, USA: IEEE Press, Jan 2003.

[11] B. Przydatek; D. Song; and A. Perrig: "SIA: secure information aggregation in sensor networks." In: Proceedings of ACM Conference on Embedded networked sensor systems SenSys. Los Angeles, USA: ACM Press, Nov 2003, pp. 255-265, ISBN 1-58113-707-9.

[12] S. Zhu; S. Setia; S. Jajodia; and P. Ning: "An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks." In: Proceedings of IEEE Symposium on Security and Privacy. Oakland, USA: IEEE Press, 2004, ISBN 0-7695-2136-3.

[13] L. Buttyán; P. Schaffer; and I. Vajda: "RANBAR: RANSAC-based resilient aggregation in sensor networks." In: Proceedings of Workshop on Security of Ad Hoc and Sensor Networks. Alexandria, USA: ACM Press, Oct 2006, pp. 83-90, ISBN 1-59593-554-1.

[14] H.C. van Tilborg, Ed.: Encyclopedia of Cryptography and Security. Springer, 2005, ISBN 038723473X.

[15] UCLA Parallel Computing Laboratory: "About GloMoSim," http://pcl.cs.ucla.edu/projects/glomosim/, Jul 2006.

[16] J. Wilke; E.-O. Blaß; and M. Zitterbart: "ESAWN-NR: Authentic Aggregation and Non-Repudiation in Wireless Sensor Networks." Kanazawa, Japan: Fifth International Conference on Networked Sensing Systems (INSS 2008), Jun. 2008, p. 254, ISBN 978-4-907764-31-9.

[17] E.-O. Blaß; L. Tiede; and M. Zitterbart: "An Energy-Efficient and Reliable Mechanism for Data Transport in Wireless Sensor Networks." In: Third International Conference on Networked Sensing Systems (INSS), Chicago, USA, May 2006, pp. 211-216, ISBN 0-9743611-3-5.

[18] Crossbow Technology Incorporated: "Radio, rf conecpts and tos radio stack," http://www.eol.ucar.edu/isf/facilities/isa/internal/CrossBow.

[19] Atmel Cooperation: "ATmega128(L) reference," http://www.atmel.com/dyn/resources/prod documents/doc2467.pdf, 10 2006, rev. 2467O.

[20] D. Westhoff; J. Girao; and M. Acharya: "Concealed data aggregation for reverse multicast traffic in sensor networks: Encryption, key distribution, and routing adaptation," IEEE Transactions on Mobile Computing, vol. 5, no. 10, pp. 1417-1431, Oct 2006, ISSN: 1536-1233.

[21] D. Wagner: "Cryptanalysis of an algebraic privacy homomorphism." In: Proceedings of Information Security Conference ISC, Bristol, UK, Oct 2003, pp. 234-239, ISBN 3-540-20176-9.

[22] H. Chan; A. Perrig; and D. Song: "Secure hierarchical in-network aggregation in sensor networks." In: Proceedings of Conference on Computer and Communications Security, Alexandria, USA, Nov 2006, pp. 278-287, ISBN 1-59593-518.

[23] Y. Yang; X. Wang; S. Zhu; and G. Cao: "SDAP: A secure hop-by-hop data aggregation protocol for sensor networks." In: Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing, Florence, Italy, 2006, pp. 356-367, ISBN 1-59593-368-9.

[24] T.K. Srikanth and S. Toueg: "Simulating authenticated broadcasts to derive simple fault-tolerant algorithms," Distributed Computing, vol. 2, no. 2, pp. 80-94, 1987.

[25] M.N. Garofalakis; J.M. Hellerstein; and P. Maniatis: "Proof sketches: Verifiable in-network aggregation." In: Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, Istanbul, Turkey, Apr. 2007, pp. 996-1005.