Relaxed Authenticity for Data Aggregation in Wireless Sensor Networks

Erik-Oliver Blaß Joachim Wilke Martina Zitterbart
Institute of Telematics, Universität Karlsruhe (TH)
Germany
{blass|wilke|zit}@tm.uka.de

ABSTRACT

In-network data aggregation allows energy-efficient communication within a sensor network. However, such data aggregation introduces new security challenges. As sensor nodes are prone to node-*compromise*, a fraction of nodes might act maliciously and forge aggregated data. For arbitrary aggregation functions, the verification of *authenticity* of aggregated data, i.e., its correctness, integrity, and origin, is impossible. Thus, one can either aggregate data and save energy or verify authenticity, not both. We present "ESAWN", a protocol that *probabilistically relaxes* authenticity in the presence of a fraction of compromised nodes. This enables a *trade-off* between probabilistic authenticity and probabilistic, energy-saving data aggregation. Besides theoretical analysis, we present MICA2-based simulation results. They indicate that even for high probabilities of authenticity and fraction of compromised nodes, ESAWN is more energy-efficient compared to (100%-)secure but non-aggregating communication. For example, with a fraction of 20% compromised nodes and 90% authenticity, ESAWN saves up to 40% energy.

Categories and Subject Descriptors

C.2.0 [**Computer Systems Organization**]: Computer-Communication Networks—*Security and Protection*

General Terms

Security, Algorithms

Keywords

Wireless Sensor Networks, Data Aggregation, Security, Authenticity, Probabilistic Security, Security-Energy Trade-Off

1. INTRODUCTION

In order to reduce the total number of radio transmissions and to save energy in a wireless sensor network, measured data transported towards a sink is usually aggregated [16]: Each sensor sends its measurement to an *aggregation node* that collects measurements from various nodes, computes a single *aggregate*, e.g., a summary or digest from this data, and sends the aggregate further towards the sink. Figure 1 shows an example for data aggregation: For fire detection, sensors monitor temperatures in a building consisting of two rooms. In room 1, sensors a and b measure the temperature at different positions. Nodes a and b send their data, i.e., their temperature measurements, to aggregation node x. Node x computes the average of both temperatures and, if the average is above a certain threshold, forwards the aggregate "1", a single bit representing *fire*, towards the sink. Otherwise, x forwards "0", *no fire*. Likewise, nodes c and d measure the temperature in room 2 and send their data to aggregation node y. Aggregation can *cascade*, i.e., multiple aggregations can take place consecutively, as Fig. 1 shows: if aggregation node z receives at least one "1", it forwards the aggregate "1" to the sink, which is now aware of a fire and might call for help. Aggregation forms a hierarchy, an *aggregation-tree*, with the sink-node being the root. This data transport paradigm of *in-network processing* or *data-fusion* saves energy due to fewer radio transmissions required in comparison to the naive scheme, where all measuring sensor nodes, e.g., a, b, c, d, separately send their individual measurements to the sink using traditional multi-hop communication, cf. [16, 19, 21].

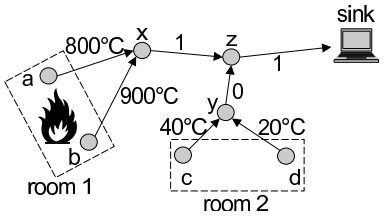


Figure 1: Example for tree-like data aggregation

However, aggregation imposes a new security problem. An attacker might physically access a certain percentage of sensor nodes, read out all their memory contents, and compromise or re-program them [3,17]. Compromised sensor nodes might act maliciously and forge an aggregate. Therefore, the sink, which is receiving an aggregate from z, must verify the authenticity of this aggregate.

Verification of authenticity is difficult to implement efficiently in wireless sensor networks: In this paper, it will be shown that, in the presence of compromised nodes, no *honest* (non-compromised) node can verify authenticity of a

SecureComm 2008 September 22 – 25, 2008, Istanbul, Turkey

received aggregate without additional data from nodes contributing to this aggregate. Yet, sending additional data to the sink foils the idea of efficient, energy-saving aggregation, where only an aggregate is forwarded to upper levels in the aggregation-tree.

This paper presents a new protocol, ESAWN (Extended Secure Aggregation for Wireless sensor Networks), a protocol for authentic, yet efficient data aggregation in wireless sensor networks. Its main idea is probabilistic verification of authenticity. ESAWN probabilistically and partially suspends aggregation within the aggregation-tree to enable authenticity verifications. This provides *probabilistically relaxed* authenticity that is weaker compared to classical, "strong" authenticity, where the user can always decide whether a received information is either authentic or not. However, probabilistically relaxed authenticity in the presence of a certain fraction of compromised nodes is much more *energy efficient* compared to classical authenticity. This approach allows the user of a sensor network to specify a trade-off between probabilistically relaxed authenticity and its energy consumption. Building on ideas published in an extended abstract [6], ESAWN's main contributions are:

- In contrast to previous work, e.g., [1, 10, 11, 22, 29, 30], ESAWN supports *arbitrary* aggregation functions and is not limited to trivial, linear arithmetic (e.g., average, "+", or "*"). Neither multiple sensors monitoring the same phenomenon nor stochastic correlations between measurements are required.
- ESAWN provides authenticity against a fraction β of compromised nodes. The user of a sensor network can parameterize ESAWN with β. The larger β, the more energy is consumed by ESAWN.
- Furthermore, the user can parameterize ESAWN with a percentage $\mathcal{P}$. Parameter $\mathcal{P}$ defines the percentage of authentic aggregates finally received at the sink, i.e., a percentage $\mathcal{P}$ of all aggregates received at the sink is authentic. Again, the larger the user demands $\mathcal{P}$ to be, the more energy is consumed by ESAWN.
- With n being the total number of nodes in the sensor network, the total energy consumption necessary due to radio transmissions and cryptographic operations is asymptotically bound to $O(n)$, i.e., per node energy consumption is bound to $O(1)$.
- All data is transported confidentially, i.e., as long as no compromised node is participating in an aggregation, the attacker will not gain any information regarding the transported data.
- Finally, as asymmetric or public-key cryptography is often assumed to be too expensive in terms of computing time and energy dissipation, ESAWN is completely based on efficient symmetric-key cryptography.

The remainder of this paper is structured as follows: While Section 2 formally defines aggregation in sensor networks, Section 3 describes the meaning of authenticity for aggregated data and points out the problem to ensure authentic aggregation. In Section 4, the supposed attacker model as well as general network assumptions are given. Sections 5 and 6 explain the basic ESAWN protocol and the probabilistic relaxation of authenticity requirements to save energy. Section 7 presents ESAWN's energy analysis and simulation results using GloMoSim. After an overview of related work in Section 8, Section 9 concludes this paper.

2. DEFINITIONS

Following Fig. 1, an *aggregation-tree* is a connected graph $G(V, E)$, with vertexes V, $|V| = n$, directed edges E, and no cycles. V represents the sensor nodes in a sensor network. Each node in V has a unique "name" or "ID". The root of G represents the sink.

Leaf nodes in G are nodes without incoming edges. Leaves, e.g., $\mathcal{S} = \{a, b, c, d\}$, are sensors measuring environmental data A, B, C, D, and sending it towards the sink using aggregation.

An edge $\overrightarrow{xz}$ between two vertexes x, z represents an *aggregation relation* between x and z: node z is called *aggregation node* for all nodes $x \in V$ with $\overrightarrow{xz} \in E$. Nodes x are called the *source nodes* for aggregation node z. An aggregation node z computes an *aggregate* agg_z using an arbitrary aggregation function F_z on source node's $x, y, \ldots$ input data, e.g., $X, Y, \ldots$: $agg_z = F_z(X, Y, \ldots)$. Again, it is important to point out that aggregation may *cascade*, i.e., as in Fig. 1, source nodes x, y for aggregation node z are also aggregation nodes for source nodes a, b and c, d, respectively. The set of aggregation nodes is $\mathcal{A}$. If an aggregation node z is also measuring data, this is represented in G by adding a new (virtual) leaf z_{leaf} to $\mathcal{S}$ and a new edge $\overrightarrow{z_{\text{leaf}}z}$ to E.

For all nodes V holds $V = \mathcal{S} \cup \mathcal{A}$.

A *path* $P \subset V$ is a sequence of nodes $P = \{x_1, x_2, \ldots, x_l\}$ with $\forall i \in \{1, \ldots, l-1\} : \overrightarrow{x_i x_{i+1}} \in E$. The *length* of a path is $|P| = l$.

For a path $P = \{x_1, x_2, \ldots, x_l\}$, node x_{i+1} is called the 1-*ancestor* of x_i in G, i.e., the *parent-node*. The k-*ancestor* of a node x_i is the $(k-1)$-ancestor of the 1-ancestor of x_i.

Furthermore, an *aggregation path* $\mathbb{P}_x$ is the set of nodes that holds $\mathbb{P}_x = \{x_i \in V \setminus \{x\} | \exists \text{ path } P \subset V : x \in P \wedge x_i \in P\}$. In other words, $\mathbb{P}_x$ consists of all ancestors of x as well as all *descendants* (children, grandchildren, ...) of x. For example in Fig. 1, $\mathbb{P}_a = \{x, z, \text{sink}\}$ or $\mathbb{P}_z = \{a, b, c, d, x, y, \text{sink}\}$.

Nodes $x_1, x_2, \ldots, x_l \in \mathbb{P}_x$ lie *in a row* on $\mathbb{P}_x$, if there is a path $P = \{x_1, x_2, \ldots, x_l\}$ in G.

For every aggregation node x, the number of source nodes is called δ_x. The arithmetic mean δ of all aggregations nodes' δ_x is $\delta = \frac{\sum_{x \in \mathcal{A}} \delta_x}{|\mathcal{A}|}$.

The *height* h of an aggregation-tree is the length of the longest path in G.

3. AUTHENTICITY VS. AGGREGATION

3.1 Authentic Aggregation

If some node z receives an aggregate agg_x from an aggregation node x, then agg_x's authenticity is made up of two important properties:

Property 1) "Correct Data":
Aggregation node x must have aggregated *correctly*, i.e., aggregate agg_x is correct against the data it depends on. For example in Fig. 1, the threshold-temperature for aggregation node x to send "1" tree-upwards is $500°C$. The non-trivial, non-linear aggregation function $F_x(A, B)$ of node x is defined as "1", if $\frac{A+B}{2} \geq 500°C$, and "0" otherwise. So, with

$800°C$ from a and $900°C$ from b, x must send $\text{agg}_x = 1$ to z – and not a forged $\text{agg}'_x = 0$.

Property II) "Correct Nodes":
All nodes involved in the aggregation, i.e., measuring nodes and aggregation nodes, must be the nodes that are expected to take part with respect to their position in the aggregation tree. For example, if z is expected to receive aggregates from x and y, it must be able to decide if a received aggregated agg_x is originated from x or y and not from different nodes like x' or y'. Note: If nodes such as x or y are compromised nodes behaving maliciously, another compromised node x' can always disguise itself and send data to z which indistinguishably appears to be from x or y . For more details, see Section 4.1.

If *every* aggregate generated in the aggregation tree is verified against these two properties, then the sink can be sure that the aggregate it finally receives is authentic: It is that aggregate computed out of the measured data. There has been no forgery.

3.2 Conflict between aggregation and authenticity

Verification of an aggregate's authenticity is difficult to realize. According to Fig. 1, node z only receives the aggregate $\text{agg}_x = 1$ from x. Assuming an *arbitrary* aggregation function $F_x(A, B)$ for node x, then z can not verify $F_x(A, B) = \text{agg}_x$ without additional information from source nodes a, b contributing to agg_x. As node x could be compromised and forge agg_x, the only way for z to verify agg_x is to obtain that data which agg_x depends on: A, B from nodes a and b. This becomes even more difficult, if you imagine more than 1 node being compromised in a row on a path within the aggregation tree: For example, if $k = 2$ nodes are possibly compromised in a row, like x and z, the sink requires data not only from x and y, i.e., agg_x and agg_y, but also A, B, C, D from a, b, c, d to verify the authenticity of a received aggregate agg_z. The sink requires data from all nodes contributing to agg_z on $k = 2$ levels below z within the aggregation tree: x, y on level 1 below z and a, b, c, d on level 2 below z. Yet, sending this additional data abolishes the idea of energy-saving due to aggregation. This means that, in the presence of compromised nodes, you can *either* save energy by data aggregation *or* spend energy to verify the authenticity of data. ESAWN's basic idea is a user-customizable trade-off between both.

4. ASSUMPTIONS

4.1 Attacker Model

Tiny sensor hardware is often deployed in public places and, for financial reasons, assumed not to be tamper-proof. Therefore, an attacker might compromise some nodes in the network. The attacker reads out the nodes' memory, e.g., secret keys, and completely controls them. Following definitions from [13], the attacker is assumed to be *static*: from all n nodes within the network, he will choose a set $\mathcal{B}$ of nodes for compromise. He chooses the nodes prior to any protocol execution and chooses them, for simplicity, all *at once* and only *one single time*. As aggregation paths in G might change from time to time, and as the attacker does not know in advance which nodes are important or not, he chooses nodes completely *randomly* in the network. He is not bound to a certain physical region of the network, but can compromise *globally*. The consequences of an attacker being able to compromise nodes not randomly but *selectively* are discussed in Section 5.3. Node compromise consumes an attacker's resources, for example, time and money, cf. [3,17]. Therefore, $|\mathcal{B}|$ is limited and *only* depending on the capabilities and resources the attacker is able or willing to spend for node compromise. Furthermore, if the attacker could compromise all sensor nodes in the network, there would be no meaningful security. So, in this paper $|\mathcal{B}| < n$ holds, and the percentage or fraction of compromised nodes in the network is called $\beta, \beta = \frac{|\mathcal{B}|}{n} < 100\%$. The sink is assumed to be the only node in the network the user has access to, e.g., to submit user-queries to the network and to receive aggregated results. If the sink could be compromised, there would be no meaningful authenticity verification possible at the sink. The sink is therefore assumed not to be compromised.

A protocol running on sensor nodes can never verify the correctness of extrinsic, measured data. For example, an attacker could forge a sensor's temperature reading simply by putting a lighter next to the sensor hardware. Resulting temperature readings can *only* be verified, if there are multiple nodes measuring the same phenomenon, and if there are stochastic correlations of measured data available. However, requiring multiple nodes for measuring a single phenomenon is unrealistic in many scenarios. Therefore and for simplicity, we assume leaves in the aggregation tree to report their *correct* measurements tree-upwards. This assumption is in accordance with related work, e.g., [1, 10, 11, 18, 22, 29–31]. Yet, if stochastic correlations are available due to multiple sensors measuring the same phenomenon, *our* scheme can be easily extended by RANBAR/RANSAC [8] techniques. As represented by adding another leaf to the aggregation tree, the same holds for aggregation nodes taking their own measurements into account. In accordance to related work, we assume for simplicity that aggregation nodes use their correct measurements. Otherwise, RANBAR/RANSAC techniques would again be required.

We assume all compromised nodes to know each other and cooperate in their forgery. If they want to, compromised nodes may behave just like honest nodes; however, they can also behave non-compliant, i.e., to forge an aggregate. Using out-of-band mechanisms, all nodes share the same knowledge all the time. It is impossible for an honest node to decide whether another node is compromised or not. Compromised nodes are *active*, i.e., they will not only listen to wireless traffic but are able to generate messages.

The attacker is assumed not to be *destructive* with his attacks. Instead, the attacker is interested in gaining more sophisticated *incentives*. For example, he will not launch Denial-of-Service (DoS) attacks, but tries to convince the sink of a forged temperature in, e.g., a building. More general, the goal of the attacker is to successfully forge at least one aggregate unnoticed by any possible verification procedures. Therefore, ESAWN's goal is to ensure that every single honest node will recognize a received but forged aggregate and that such malicious behavior is revealed to the sink.

4.2 Network Assumptions

In this paper, the structure of an aggregation-tree G is assumed to be given, e.g., by an appropriate mechanism like Directed Diffusion [19]. As a result, every node in G knows

its current next-hop towards the sink.

For simplicity, each aggregation is assumed to take place "synchronized". All nodes involved in a single aggregation, i.e., the aggregation node z, the source nodes of z, and possible verification nodes of z, synchronously know *when* data has to be sent and aggregated.

Wireless communication is typically prone to transmission errors and packet loss and therefore unreliable. Reliable communication, e.g., based on "Stop-and-Wait", is expensive in Sensor Networks as shown in [5]. Nevertheless in this paper, robust communication is assumed, e.g., using a mechanism like "Stop-and-Wait" as provided by TinyOS [25]. Future work will study the use of more efficient and appropriate reliability-mechanism together with ESAWN.

Furthermore, we assume every node to share different pairwise secret symmetric keys with its $(k+1)$ ancestors in G. Such keys are typically exchanged in advance using an arbitrary key exchange protocol, e.g., [7, 15]. Also, if G changes from time to time, new keying is easily done by the same mechanism [7]. Here, integer k is assumed to be a public, well-known system parameter, which might be distributed to every sensor during deployment. Every node in the network, maybe even the attacker, knows k. Section 5.3 discusses, how k is chosen. Sharing keys with the $(k+1)$ ancestors in G implies that a node also shares keys with its descendants on all $(k+1)$ levels tree-downwards in G.

Moreover, this trivially implies that every node knows these ancestors and descendants: Every node x has the local knowledge about its aggregation path $\mathbb{P}_x$ on the $(k+1)$ levels and tree-downwards. Although $\mathbb{P}_x$ might change from time to time, it is reasonable to assume x to know this subset of $\mathbb{P}_x$, e.g., by using an extension of [19].

We assume the number of source nodes per aggregation node to be randomly distributed with a mean of $\delta \geq 1$. This means that on average, every aggregation node z computes its aggregate out of δ-many source nodes $x_1, \ldots, x_\delta$'s input $X_1, \ldots, X_\delta$, i.e., $\text{agg}_z = F_z(X_1, \ldots, X_\delta)$. Thus, for the total number of nodes approximately holds

$$n = \sum_{i=0}^{h} \delta^i = \frac{\delta^{h+1} - 1}{\delta - 1},$$

and therefore $h = \log_\delta(1 + (\delta - 1)n) - 1$.

Again, it is important to point out that in contrast to related work, neither multiple sensors monitoring the same phenomenon nor any stochastic correlations between measurements are assumed.

5. ESAWN

Overview: The basic idea of ESAWN is to verify every aggregate computed by an aggregation node x by the k ancestors of x with respect to the authenticity properties of Sec. 3.1. For example in Fig. 1 with $k = 1$, nodes a, b, c, d do not only send their measurements to x and y, but also to z. Therewith, node z can verify x's and y's aggregates. Also, x and y do not only send their aggregates to z, but also to the sink. Likewise, the sink can verify z's aggregate. The idea is to simply *omit* aggregation on k levels in the aggregation-tree.

Here, k is a user configurable parameter which is discussed in Section 5.3. As long as there are no more than k compromised nodes in a row on any aggregation path $\mathbb{P}$, ESAWN guarantees that every honest node can detect forged aggregates. As soon as a node detects a forgery, it raises an alarm and, recursively, notifies all its $(k+1)$ ancestors in G.

5.1 Detailed Protocol Description

Figure 2 shows a simplified aggregation. Node a is a source node for aggregation node x and thereby sends its data A to x. Node x computes an aggregate agg_x and sends it to z_1. For simplicity and as indicated in Fig. 2, in the following $\delta = 1$ holds: $\text{agg}_x = F_x(A)$, $\text{agg}_{z_1} = F_{z_1}(\text{agg}_x)$, $\text{agg}_{z_2} = F_{z_2}(\text{agg}_{z_1}), \ldots$.

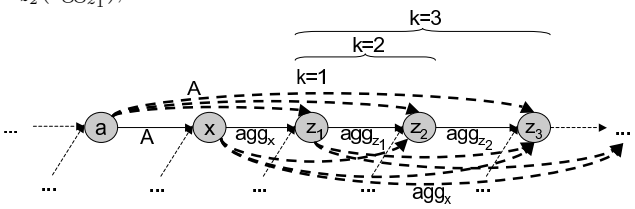


Figure 2: Simplified, typical aggregation

Now with ESAWN, node a does not only send A to x, but to $\{x, z_1, z_2, \ldots, z_k\}$, for a given k.

In the same manner, x does not only send its aggregate $F_x(A) = \text{agg}_x$ to z_1, but to $\{z_1, z_2, \ldots, z_{(k+1)}\}$.

In general, every node sends its data not only to its 1-ancestor within G, but to all $(k+1)$ ancestors.

Authenticated Encryption:

Every message is encrypted and authenticated before its transmission using pairwise keys. As assumed in Section 4, these keys have been exchanged in advance using any key exchange protocol, e.g., [7, 15]. For example, if a sends A to $\{x, z_1\}$ for $k = 1$, it sends $E_{K_{a,x}}(A)$ to x and $E_{K_{a,z_1}}(A)$ to z_1. Here, $E_{K_{x_1,x_2}}(A)$ denotes the ciphertext resulting from *authentically encrypting* A with a secret pairwise symmetric key K_{x_1,x_2} known only to x_1 and x_2. Authenticated encryption techniques using pure symmetric keys are, e.g., "Encrypt&HMAC" or the use of redundancy, cf. [4, 24, 27]. Note: Authenticated encryption does not only provide data authenticity, but also data confidentiality. While confidentiality is not in the main focus of this work, confidentiality is often desirable against pure eavesdropping attackers and can be realized with negligible additional overhead, see [24]. Therefore in this paper, authenticated encryption is used as a building block, yet it can be replaced by, e.g., pure HMAC techniques.

Multi-Hop Communication:

In general, all communication is multi-hop in ESAWN. For example, if a wants to send $E_{K_{a,z_1}}(A)$ to z_1, node a sends $E_{K_{a,z_1}}(A)$ to x which forwards it to z_1. This is possible, because a knows z_1 being the parent-node of x, cf., Section 4.2. Here, a malicious node x might deny forwarding $E_{K_{a,z_1}}(A)$ to z_1 and simply drop $E_{K_{a,z_1}}(A)$. However, not forwarding but dropping data is a DoS-attack which can be achieved much more easily in sensor networks, e.g., by jamming all communication channels. As assumed in Section 4.1, the attacker is not destructive, but interested in gaining incentives with his attack.

5.2 Authenticity Verification

With ESAWN, every node receiving data verifies this data against two authenticity-properties: I) "Correct Data" and II) "Correct Nodes".

I) Correct Data

Each node verifying an aggregate, the *verifier* node, does not only receive an aggregate agg from an aggregation node but also data from the source nodes of the aggregation node which agg is based on. Using this source data, the verifier recomputes an aggregate agg$'$ and compares agg$'$ to the received agg. If both match, the verifier has successfully verified property I) of the aggregation.

For example in Fig. 2, verifier z_1 is verifying aggregation node x's aggregation. Node z_1 does not only receive agg_x, but with $k = 1$ also A from source node a which agg_x is based on. If z_1 is honest, it computes an aggregate agg'_x from received data A using the same aggregation function F_x as x: $\text{agg}'_x = F_x(A)$. As soon as z_1 is receiving agg_x from x, it compares agg'_x with agg_x. If both aggregates match, z_1 has successfully verified property I) of x's aggregation in the presence of at most $k = 1$ malicious node in a row. Otherwise, if the aggregates do not match, *either* x is compromised and forging its aggregate *or* a is compromised and provides z_1 with false input data A. In this case, the verification has failed, z_1 is aware of malicious behavior regarding x's aggregation and will "raise an alarm": using multi-hop communication as explained above, z_1 sends $E_{K_{z_1,z_{1+i}}}(\text{ALARM})$, $1 \leq i \leq k+1$, to its $(k+1)$ ancestors. In addition, every node receiving an ALARM also sends ALARM to its $(k+1)$ ancestors. As one of z_1's $(k+1)$ ancestors, e.g., z_2, might also be malicious, z_2 could refuse to forward the ALARM messages. However, as all of z_1's ancestors expect a message from z_1, *either* z_1's aggregate *or* an alarm message, a missing message is conspicuous. So, after a timeout, e.g., z_3 will itself generate an ALARM message. As a result, the sink will eventually receive an ALARM message and is therefore aware of malicious behavior within the sensor network. This scheme allows compromised nodes to possibly raise "false-alarms". However, the attacker wants to remain unnoticed by the sink and does not want to draw attention to any malicious behavior going on in the network. As stated in Section 4.1, the attacker's goal are incentives, e.g., making an honest node "accept" a forged aggregate unnoticed.

In a situation with $k = 2$ compromised nodes in a row and verifier z_2, z_2 does not only receive agg_{z_1}, but also agg_x and A. Again assuming that z_2 is honest, it computes $\text{agg}'_x = F_x(A)$, $\text{agg}'_{z_1} = F(\text{agg}'_x)$, and $\text{agg}''_{z_1} = F(\text{agg}_x)$. If $\text{agg}_{z_1} = \text{agg}_x = \text{agg}'_x$ and $\text{agg}'_{z_1} = \text{agg}''_{z_1}$, then z_2 has successfully verified property I) for the aggregations of nodes x and z_1. Otherwise, z_2 presumes malicious behavior and raises an alarm. With a total of $k = 2$ malicious nodes in a row, nodes $\{a, x\}$, $\{x, z_1\}$ could have been compromised. Note: the situation where $\{a, z_1\}$ are compromised with x honest is equivalent to $k = 1$. As x is honest, it will not try to forge an aggregate. The aggregate of compromised node z_1 can be verified by z_2 with $k = 1$.

As ESAWN's correctness regarding authenticity property I) is not quite intuitive, this property will be formally proved.

5.2.1 Formal proof of property I) "Correct Data"

In the following, consider all the nodes of a tree G to be colored. Every node is either *white* or *grey*.

Definition: The $(k+1)$-*Ancestor-Closure*, $(k+1)$-AC, of a tree G is a tree that extends G, $G \subseteq (k+1)$-AC. $(k+1)-$AC extends G by adding additional edges from every node to its 1-ancestor, 2-ancestor, ..., $(k+1)$-ancestor.

Definition: The *white-subtree* (WS) of G is the subtree of G consisting only of the white nodes of G as well as the edges of G between white nodes.

Definition: A tree G is *root-connected*, if there is a path from every node to the root of G.

Assumption A_k: On every path in a colored G with $(k+1)$ nodes lying in a row, at least one them is white.

Proposition

$A_k \Rightarrow$ if the root w of G is white, then the WS of $(k+1)$-AC of G is root-connected.

Proof

By *structural induction* on trees consisting of subtrees.

Basis

Consider a tree consisting of only one node, i.e., the root w. $A_k \Rightarrow$ if root w is white, then the WS of $(k+1)$-AC of w is trivially root-connected.□

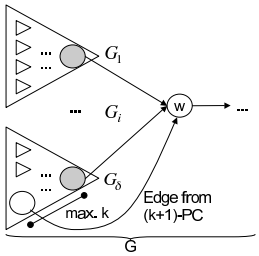


Figure 3: δ-subtrees of G

Induction Hypothesis

Given are δ trees $G_1, \ldots, G_\delta$. The proposition holds for all G_i.

Inductive Step

Consider a $(k+1)$-AC consisting of δ-subtrees and a new root w as of Fig. 3.

$A_k \Rightarrow$ if w is white then

1. if a root of any subtree is white, then WS of $(k+1)$-AC is connected to w – the induction hypothesis holds for this subtree.□

2. if a root w' of any subtree is grey, then this subtree is composed of smaller subtrees that hold the proposition. As of assumption A_k, the *next* white node is maximum k-nodes away from w'. Finally, $(k+1)$-AC implies the existence of an edge between this white node and w, cf. Fig. 3.□

Conclusion: The grey nodes represent the compromised nodes, the white represent the honest nodes. Assumption A_k is the assumption regarding the attacker, i.e., there are no more than k compromised nodes in a row taking part in a verification. Therefore, the property of the WS being root-connected means that every honest node verifying an aggregation receives data from another honest node for that aggregation. Using this data, the honest node can always detect a possible forgery and verify property I) "Correct Data".

II) Correct Nodes

As k is well-known and every node x knows its aggregation path $\mathbb{P}_x$, x expects data from its descendants on the $(k+1)$ levels below x in G. Thus, every node receiving data can

trivially verify property II) using its pairwise symmetric keys and authenticated encryption techniques.

This scheme confidentially transports aggregated data towards the sink, as long as there is no compromised node participating in the verification of an aggregate.

As a result, in the presence of up to k compromised nodes in a row per aggregation path, ESAWN guarantees that every honest node in the network detects malicious behavior regarding authenticity properties. As the sink is assumed to be honest, the user will become aware of malicious behavior and can take countermeasures.

5.3 Selecting k

ESAWN's security heavily depends on a good choice of k. If the user selects a small k such that there are $k' > k$ compromised nodes on at least one aggregation path, these compromised nodes will successfully forge an aggregate and violate authenticity properties unnoticed. So, on the one hand, the user should select k to be very large to cope with a lot of compromised nodes in the network, i.e., a high β. On the other hand, a larger k implies higher energy consumption: As shown in Section 5.1, the larger k the more network traffic and en-/decryptions result. Thus, selecting k is complicated.

However, if the user can estimate the maximum number n of nodes that will *ever* join the sensor network (a maximum number) and, in addition, estimate a mean δ, i.e., a "typical" number of source nodes per aggregation in his network, he can compute a lower bound for k. Under the assumption that a fraction β of the nodes in the network is compromised, on average no more than $\beta \cdot h$ nodes are compromised in a row on every path in G. Each node on a path, besides the sink, is compromised with probability β. Consequently, the user can trade off security against energy-consumption by selecting $k \geq \lceil \beta \cdot h \rceil$. The more compromised nodes the user wants his data transport to be secure against, the more he has to pay in terms of energy.

A stronger attacker than the one assumed in this paper might compromise nodes not randomly but selectively. He might choose $|\mathcal{B}|$ nodes in a row in G. As a result, the user would have to set k to $k = |\mathcal{B}|$. However, as $k = |\mathcal{B}|$ is considerably larger compared to, e.g., $k = \lceil \beta \cdot h \rceil$, this will result in very high energy consumption, cf. Section 7.2.

6. PROBABILISTIC RELAXATION

It is quite obvious that basic ESAWN consumes more energy than aggregation without any security due to the increased number of data transmissions necessary with $k = 1, 2, 3, \ldots$. To cope with the increased energy consumption, ESAWN allows to *probabilistically* verify aggregations.

Similar to k, assume another public, well-known system parameter $0 < p \leq 1$. Aggregation nodes are verified independently from each other with the same probability p. This is done as follows:

6.1 Distributing a Seed

First of all, all nodes taking part in verification of an aggregation node x require the same seed or internal state for a pseudorandom number generator PRNG [27].

Because k is well-known, every child of an aggregation node x as well as the k ancestors of x know that they have to take part in verification of x. Therefore, they utilize, e.g., $H(x)$, a hash value computed from the unique ID x with a well-known hash-function, as a *seed* for use in PRNG: seed $:= H(x)$. Thus, all nodes taking part in the verification of x are aware of the same seed and use it as the initial internal state for the PRNG.

6.2 Random Verification

Every time, x computes and forwards a new aggregate

- all children of x, the k ancestors of x, and x itself pick the same pseudorandom number r, $0 \leq r \leq 1$ using PRNG with the same internal state.
- If $r \leq p$, basic ESAWN verification is executed as described in Section 5.1.
- Otherwise, x just forwards its aggregate to its 1-ancestor in G.
- All children, the k ancestors, and x itself necessarily update their PRNG's internal state [27].

Here, the assumption that nodes have synchronized knowledge of an aggregation taking place is important. Without it, a secure protocol for notifying nodes that an aggregation is going to take place would be necessary.

Random verification results in a probabilistically relaxed authenticity, namely authenticity of each aggregate with probability p. The higher the user selects p, the more often ESAWN is executed – this results in higher security but more energy consumption.

As at most k nodes in a row are compromised, the attacker possibly also knows the seed. It is important to note that this is *not* a security problem: Although the attacker will know, *when* successful forgery is possible, he can not *control* p or *whether* an actual aggregation from x is verified. The attacker can still only successfully forge aggregates with probability $(1 - p)$.

6.3 Defining $\mathcal{P}$

The user of the sensor network is most likely not only interested in the authenticity of each individual aggregation node, but he is interested in the probability that the "final" aggregate, received at the sink, is authentic. Given that each individual aggregation is statistically independently authentic with probability p, how can the probability $\mathcal{P}$ of an authentic aggregate received at the sink be computed? In return, how can p be computed out of a given $\mathcal{P}$? This helps to parameterize ESAWN regarding a certain user demand for a probability $\mathcal{P}$.

The probability that "an aggregation node is *either* honest, *or* if it is compromised, it must be verified" can be expressed as: $(1 - \beta + \beta \cdot p)$. As each aggregation node is verified statistically independently, $\mathcal{P} = (1 - \beta + \beta \cdot p)^{|\mathcal{A}|}$ holds, with $\mathcal{A}$ the set of aggregation nodes. On every level i in G, there are δ^i aggregation nodes. With the assumed definition of the height of an aggregation tree, the total number of aggregation nodes can be computed as

$$|\mathcal{A}| = \sum_{i=1}^{h-1} \delta^i = \sum_{i=1}^{\log_\delta(1+(\delta-1)n)-2} \delta^i = \frac{n-1}{\delta} - 1.$$

Therewith, $\mathcal{P} = (1 - \beta + \beta \cdot p)^{\frac{n-1}{\delta}-1}$, and finally

$$p = 1 - \frac{1 - \sqrt[(n-\delta-1)]{\mathcal{P}^\delta}}{\beta}.$$

As with k, the user can trade off security against energy consumption. The higher his demand for authenticity $\mathcal{P}$, the larger p will be and the more often ESAWN has to verify every aggregation. Just as well, the larger the fraction of compromised nodes β the user wants to protect his data transport against, the higher is p and the more energy is consumed. An ESAWN *configuration* (k, p) is a user determinable trade-off between authenticity and resulting energy consumption.

7. EVALUATION

Energy consumption is one of the most critical aspects in sensor networks. To ESAWN's feasibility and scalability, this section analyzes its energy consumption. Thereby, for better estimation and to put things into perspective, ESAWN is compared with two different, typical data transport schemes:

1.) *Non-Authentic Aggregation:* This is standard aggregation as proposed in [19,21]. Data is aggregated on its way towards the sink, however, without any security and authenticity. In terms of energy consumption, this is a lower bound for ESAWN. ESAWN can never be more energy-efficient as Non-Authentic Aggregation, because ESAWN is providing authenticity in addition to data aggregation.

2.) *Authentic Non-Aggregation:* In this scheme, measuring nodes authentically encrypt their measurements with pairwise-secret keys they share with the sink. Nodes within the network naively forward every received (authentically encrypted) date further towards the sink. There is no in-network processing of multiple input data or data-fusion taking place. The sink is receiving all individually authentically encrypted measurements, decrypts them and aggregates them itself. From a security perspective, Authentic Non-Aggregation yields classical, "strong" authenticity. The sink can always decide whether a received date is from a node it expects it to be from. Authentic Non-Aggregation is some kind of upper bound or baseline for ESAWN: if ESAWN or a specific ESAWN configuration (k, p) consumes more energy than Authentic Non-Aggregation, there would be no use for ESAWN – the naive Authentic Non-Aggregation scheme is more secure and more energy efficient. Hence, during evaluation, Authentic Non-Aggregation is used as a baseline. ESAWN's energy consumption is measured against the energy consumption of Authentic Non-Aggregation to see, how much energy can possibly be saved due to probabilistically relaxing authenticity requirements in the presence of a certain fraction of compromised nodes.

7.1 Upper Bounds for Energy Consumption

A *run* is a single protocol execution on the entire aggregation-tree: every leaf takes one measurement that is transported towards the sink. Depending on the protocol executed, i.e., ESAWN, Authentic Non-Aggregation, or Non-Authentic Aggregation, measurements are possibly authentically encrypted or aggregated by aggregation nodes (several times) on their way to the sink.

7.1.1 Non-Authentic Aggregation

As there is no authentic encryption or decryption with this protocol, the only energy costs arise due to wireless message transfer. During a run, every node besides the sink transfers only one message – its aggregate. Therefore, a total of $(n-1) \in O(n)$ messages are transferred. Per node energy consumption scales with $O(1)$.

7.1.2 Authentic Non-Aggregation

With this protocol, each aggregation node works as a simple forwarding node, i.e., aggregation nodes forward every received data further towards the sink. As there are on average δ^h leaves in G,

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

messages have to be sent for one run. With

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

$f(n, \delta)$ scales with $O(n \log n)$. Thus, the number of transmissions per node scales with $O(\log n)$. In addition, there are δ^h encryptions and decryptions, this scales with $O(1)$ per node. In total, energy consumption scales with $O(\log n)$ per node.

7.1.3 ESAWN

Here, every node x sends its aggregate authentically encrypted to x's 1-ancestor in G. Node x also sends k authentically encrypted aggregates to x's 1-ancestor for forwarding to the $\{2, 3, \ldots, (k+1)\}$-ancestor of x. In addition, every node x has to forward all data from the δ^i nodes on levels $i = 1, \ldots, k$ *below* x in G. For every node on level i below x in G, x has to forward $(k - i + 1)$ data, respectively. Altogether, every node sends

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

messages. With a fixed number of compromised nodes $|\mathcal{B}|$, cf. Section 4.1, this expression scales with $O(1)$ with respect to the rising total number of nodes n. Furthermore, every node x authentically encrypts his aggregate $(k+1)$-many times. Node x also receives authentically encrypted data from all his descendants on levels $1, \ldots, (k+1)$ below x in G that have to be decrypted, i.e., x decrypts

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

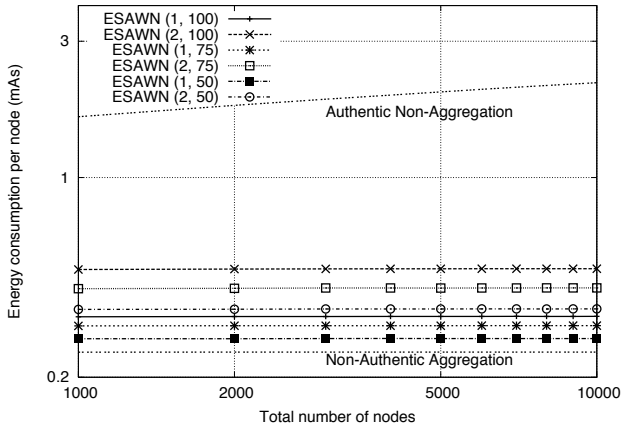
many times. With a fixed $|\mathcal{B}|$, this scales with $O(1)$ per node.

Overall, ESAWN's energy consumptions scales with $O(1)$.

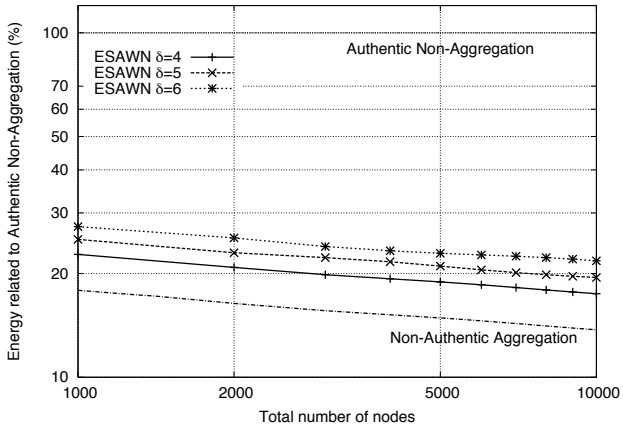
7.2 Simulation Results

To further evaluate the energy consumption for configurations (k, p), ESAWN was implemented within the discrete event simulation environment GloMoSim [26]. The energy consumption measured was made up of radio transmissions for messages that are sent during protocol execution as well as required authentic encryption and decryption operations respectively. Energy costs for radio transmissions and cryptographic operations were derived from Crossbows's popular MICA2 platform running TinyOS, cf. [25]. As a simple static routing, every node in the network had a simple routing table pre-calculated prior to each simulation. Please refer to Table 1 in the Appendix which sums up important simulation parameters used.

Aforementioned *runs* were simulated for all three protocols. Fig. 4(a) shows the total energy consumption per node



(a) Total energy consumption, varying (k, p), $\delta = 3$



(b) Relative energy consumption, varying δ, $k = 1, p = 100\%$

Figure 4: Per node energy consumption for different (k, p) and δ

that is accumulated with $n = \{1000 \dots 10000\}$ randomly placed nodes and a *normally* distributed $\delta = 3$. While Non-Authentic Aggregation and ESAWN consume a fixed amount of energy per node, regardless of n, energy consumption of Authentic Non-Aggregation rises due to the increasing number of messages that have to be authentically encrypted and sent. You can clearly see that ESAWN with (k, p) between $(1, 50)$ and $(2, 100)$ consumes less energy than Authentic Non-Aggregation. The higher k and the higher p, the higher the resulting energy consumption. Furthermore, $(2, 50)$ is more expensive than $(1, 100)$. In general, it can be observed that with $k' > k$, $(k', \frac{k}{k'}p)$ is more expensive than (k, p). In terms of energy consumption, k is more dominant than p.

To clarify ESAWN's energy savings compared to the naive, secure but non-aggregating data transport, Figures 4(b), 5(a), and 5(b) show ESAWN's *relative* energy-consumption with respect to Authentic Non-Aggregation. For example, Fig. 4(b) shows ESAWN's relative energy consumption with varying δ. Using $\delta = 4$ and $n = 1000$ nodes, ESAWN consumes less than 30% energy compared to Authentic Non-Aggregation with $\delta = 4$, $n = 1000$. The higher δ, the higher ESAWN's relative energy consumption, because broader trees with a fixed n are shorter. In a shorter tree, Authentic Non-Aggregation consumes less energy, so, ESAWN's relative energy consumption is higher. Furthermore, ESAWN's relative energy consumption falls with growing n, because aggregation itself already saves more energy with larger n compared to non-aggregation.

Different values for δ result in energy consumptions running parallel to each other. In the following, δ will therefore be fixed to a value of $\delta = 4$.

Fig. 5(a) and 5(b) show the energy consumption necessary to reach a certain user demand for $\mathcal{P}$ with $\beta = 1\%$ compromised nodes, cf. Fig. 5(a), and $\beta = 20\%$, cf. Fig. 5(b). All lines in both figures are labeled with configurations (k, p) that were used to achieve *at least* $\mathcal{P}$. As p and in particular k can only be chosen discretely, i.e., using integer values, some configurations (k, p) achieve not only $\mathcal{P}$, but are also sufficient for a $\mathcal{P}' > \mathcal{P}$. For example, in Fig. 5(b), for $n = 4000$ sensor nodes, configuration $k = 4$, $p = 100\%$ achieves $\mathcal{P} = 50\%$, but is also sufficient for $\mathcal{P} = 90\%$. However, a configuration with $k = 3$ or $p = 99\%$ would not neither achieve $\mathcal{P} = 50\%$ nor $\mathcal{P} = 90\%$. Maybe, an impractical configuration of, e.g., $k = 3.72$, $p = 99.352\%$ would achieve $\mathcal{P} = 50\%$, but not $\mathcal{P} = 90\%$: The integer based configurations in Fig. 5 are the most energy-saving configurations achieving a certain $\mathcal{P}$.

These simulations indicate the immense energy savings possible, e.g., with $n = 10000$ nodes, $\beta = 1\%$, and $\mathcal{P} \geq 50\%$, up to 80% energy can be saved.

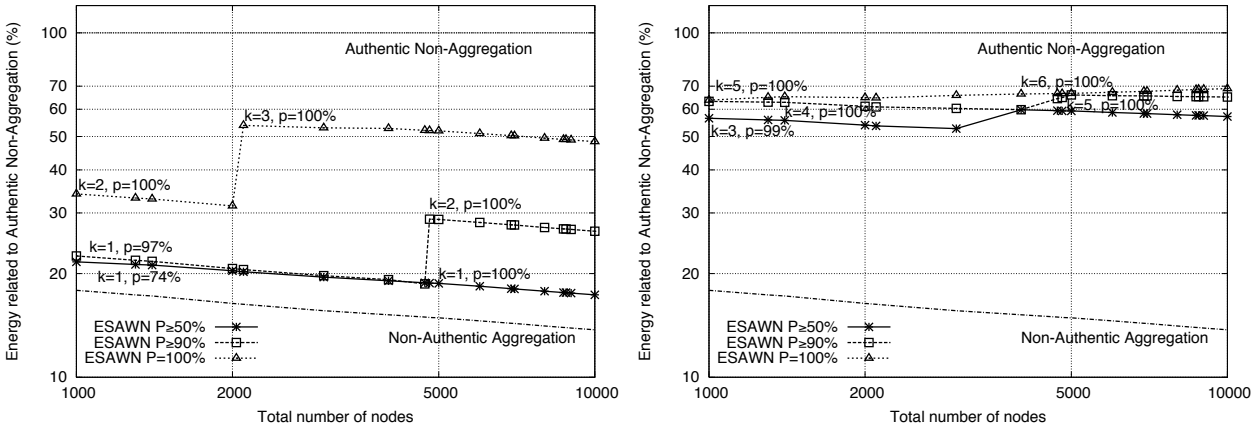
In both figures, there are "jump discontinuities", e.g., in Fig. 5(a), at $n = 2000$ nodes and $\mathcal{P} = 100\%$. As β is fixed and n growing, the total number of compromised nodes $|\mathcal{B}| = \beta \cdot n$ grows. Therefore, ESAWN has to cope with more compromised nodes, which requires higher k. As k is only discretely increased, "jump discontinuities" are the result.

Discussion: The higher the percentage of compromised nodes within the sensor network, the more energy has to be spent to reach a certain authenticity. However, as shown in Fig. 5(b), even with a high percentage of compromised nodes, $\beta = 20\%$, and a high $\mathcal{P} = 90\%$, ESAWN still saves $> 40\%$ energy for $n = 10000$ sensor nodes compared to Authentic Non-Aggregation. The more the user accepts a relaxed authenticity $\mathcal{P}$ for his aggregates received at the sink, the more energy can be saved. For example, if the user is willing to accept only $\mathcal{P} = 50\%$ authenticity, 45% of energy can be saved. Using ESAWN, the user can therefore specify a trade-off between authenticity and energy savings.

8. RELATED WORK

Some work regarding secure data aggregation is based on privacy homomorphisms, cf. [1, 10, 29]. Here, aggregation nodes can compute their aggregate from authentically encrypted measurements without decryption. However with this approach, aggregation functions are strictly limited to trivial *linear* arithmetic operations, e.g., "+" or "∗". Also, most types of comparisons are infeasible [23]. So, support of *arbitrary* aggregation functions is, contrary to this work, impossible. Furthermore, privacy homomorphisms are prone to known-plaintext, chosen-plaintext, and replay-attacks, cf. [28].

In "SIA" [22], the sink verifies a received aggregate agg by randomly asking some sensor and aggregation nodes for



(a) $\beta = 1\%$ (b) $\beta = 20\%$

Figure 5: Per node energy consumption for different β and $\mathcal{P}$

the data they sent upwards within the aggregation tree. Using this data, the sink tries to statistically reconstruct agg. However, this statistical verification of authenticity *only* supports trivial mathematical aggregation functions, such as average or median. This also holds for SIA's extension presented in [11]. Similar, SDAP [30] enables the sink to detect "outliers", i.e., aggregates that statistically differ from a mean aggregate. Again, only simple aggregation functions as "+" or mean are supported, and this approach requires a lot of nodes to monitor the same phenomenon for the detection of outliers – which might be unrealistic in some scenarios.

A protocol to filter maliciously injected data is presented in [31]. Here, $(k+1)$ nodes send the same measurement secured by HMACs to their k-ancestor, respectively, so injected data of up to k compromised nodes can be detected by majority-vote. Yet, the scheme does not support any form of aggregation, i.e., *in-network-processing* by aggregate computation from source nodes' input data, but simply forwards all measurements to the sink. Also, $(k+1)$ sensors are always required to monitor the same phenomenon. Similar to ESAWN, the scheme in [9] sends encrypted data from nodes in an aggregation tree to their predecessors. However, this only provides confidentiality and not authenticity. As there is no probabilistic relaxation in [9] as in ESAWN, high energy consumption is expected. Finally, the scheme does not support aggregation nodes taking their own measurements into account, as ESAWN does.

In [12], a probabilistic authentication mechanism is presented that limits the impact of malicious dummy messages in a wireless sensor network. The authors present a security versus energy trade-off. Still, in-network processing of data towards a sink is not addressed.

The authors of [18] use *hash-chains* for authenticity: every sensor node shares a pairwise secret with the sink that is used to construct a hash-chain. At time-period t, each measuring sensor uses the current element of its hash-chain to send a commitment of its measurement tree-upwards. At time-period $(t+1)$, the sink floods the elements of all hash-chains for period t into the network. Therewith, nodes can verify previously received commitments. However, this scheme is secure against only one compromised node, and nodes can only verify a received aggregate some time after receiving it and after using it as input for their own aggregation. For many scenarios, this is inappropriate.

9. CONCLUSION

Energy saving aggregation contradicts the verification of authenticity. You can either aggregate data and save energy, or omit aggregation and verify data authenticity and correctness.

This paper presents ESAWN, a user *customizable trade-off* between authenticity and energy consumption. In a sensor network where data is aggregated on its way towards the sink, the user can probabilistically relax authenticity in the presence of a certain fraction of compromised nodes to save energy. Aggregations are verified with probability p in the presence of a total of $|\mathcal{B}|$ compromised nodes. The result is a customizable probability $\mathcal{P}$ that an aggregate received at the sink has been authentically transported through the whole network.

This leads to a clearly weaker type of security compared to a naive non-aggregation scheme. However, ESAWN can significantly decrease energy-consumption compared to non-aggregation, i.e., from $O(\log n)$ down to $O(1)$ per node. ESAWN has been implemented for GloMoSim. Resulting simulations also exhibit the enormous energy savings possible. For example, with a total of 10000 sensor nodes, 1% compromised nodes, and $\mathcal{P} \geq 90\%$, over 70% energy can be saved.

Acknowledgments: The authors wish to thank Roland Bless, Curt Cramer, Felix Freiling, Artur Hecker, and Bernhard Hurler for their support.

10. REFERENCES

[1] M. Acharya, J. Girao, and D. Westhoff. Secure comparison of encrypted data in wireless sensor networks. In *Proceedings of IEEE Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, pages 47–53, Riva del Garda, Italy, Apr 2005. ISBN 0-7695-2267-X.

[2] Atmel Corporation. Atmel atmega128 datasheet, 2002. `http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf`.

[3] A. Becher, Z. Benenson, and M. Dornseif. Tampering with motes: Real-world physical attacks on wireless sensor networks. In *Proceedings of International Conference on Security in Pervasive Computing*, pages 104–118, York, UK, Apr 2006. ISBN 3540333762.

[4] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *Proceedings of International Conference on the Theory and Application of Cryptology and Information Security*, pages 531–545, Kyoto, Japan, Dec 2000. ISBN 3-540-41404-5.

[5] E.-O. Blaß, L. Tiede, and M. Zitterbart. An energy-efficient and reliable mechanism for data transport in wireless sensor networks. In *Proceedings of International Conference on Networked Sensing Systems*, pages 211–216, Chicago, USA, May 2006. ISBN 0-9743611-3-5.

[6] E.-O. Blaß, J. Wilke, and M. Zitterbart. A security–energy trade-off for authentic aggregation in sensor networks. In *Proceedings of IEEE Communications Society Conference on Sensor and AdHoc Communications and Networks (SECON)*, pages 135–137, Washington D.C., USA, Sep 2006.

[7] E.-O. Blaß and M. Zitterbart. An efficient key establishment scheme for secure aggregating sensor networks. In *Proceedings of ACM Symposium on Information, Computer, and Communications Security (ASIACCS)*, pages 303–310, Taipeh, Taiwan, 2005. ISBN 1-59593-272-0.

[8] 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*, pages 83–90, Alexandria, USA, Oct 2006. ISBN 1-59593-554-1.

[9] C. Castelluccia. Securing very dynamic groups and data aggregation in wireless sensor networks. In *Proceedings of Mobile Adhoc and Sensor Systems*, pages 1–9, 2007. ISBN 978-1-4244-1455-0.

[10] C. Castelluccia, E. Mykletun, and G. Tsudik. Efficient aggregation of encrypted data in wireless sensor networks. In *Proceedings of ACM/IEEE Mobiquitous*, pages 109–117, San Diego, USA, Jul 2005.

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

[12] V. Coskun, E. Cairici, A. Levi, and S. Sancak. Quarantine region scheme to mitigate spam attacks in wireless-sensor networks. *IEEE Transactions on Mobile Computing*, 5(8):1074–1086, 2006. ISSN 1536-1233.

[13] R. Cramer and I. Damgård. Introduction to secure multi-party computations. In *Contemporary Cryptology: Advanced Courses in Mathematics*, pages 41–87. Birkhauser, 2005. ISBN 3-7643-7294-X.

[14] Crossbow Inc. Radio, rf concepts, and tos radio stack, 2006. `http://www.eol.ucar.edu/isf/facilities/isa/internal/CrossBow/PresentationOverheads/Day1_Sect06_RFConcepts.pdf`.

[15] L. Eschenauer and V. Gligor. A key management scheme for distributed sensor networks. In *Proceedings of ACM Computer and Communications Security*, pages 41–47, Washington D.C. USA, Nov 2002.

[16] E. Fasolo, M. Rosso, J. Widmer, and M. Zorzi. In-network aggregation techniques for wireless sensor networks: A survey. *IEEE Wireless Communications*, 14(2):70–87, 2007. ISSN 15361284.

[17] C. Hartung, J. Balasalle, and R. Han. Node compromise in sensor networks: The need for secure systems. Technical report, Department of Computer Science, University of Colorado, USA, 2005. `http://www.cs.colorado.edu/department/publications/reports/docs/CU-CS-990-05.pdf`.

[18] L. Hu and D. Evans. Secure aggregation for wireless networks. In *Proceedings of IEEE Symposium on Applications and the Internet Workshops (SAINT)*, pages 384–391, Orlando, USA, Jan 2003.

[19] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for Wireless Sensor Networking. *ACM Transactions on Networking*, 11(1):2–16, Jan 2003. ISSN 1063-669.

[20] C. Karloff, N. Saastry, and D. Wagner. Tinysec: A link layer security architecture for wireless sensor networks. In *Proceedings of ACM Conference on Embedded Networked Sensor Systems*, pages 162–175, Baltimore, USA, Nov 2004. ISBN 1-58113-879-2.

[21] B. Krishanamachari, D. Estrin, and S. Wicker. The impact of data aggregation in wireless sensor networks. In *Proceedings of IEEE Distributed Event-Based Systems*, pages 575–578, Wien, Austria, Jul 2002.

[22] 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*, pages 255–265, Los Angeles, USA, Nov 2003. ISBN 1-58113-707-9.

[23] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pages 169–179. Academic Press, 1978. ISBN 0122103505.

[24] P. Rogaway, M. Bellare, and J.Black. OCB: A block-cipher mode of operation for efficient authenticated encryption. *ACM Transactions on Information and System Security (TISSEC)*, 6(3):365–403, 2003. ISSN 1094-9224.

[25] UC Berkeley. Tinyos, 2005. `http://www.tinyos.net/`.

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

[27] H. C. van Tilborg, editor. *Encyclopedia of Cryptography and Security*. Springer, 2005. ISBN 038723473X.

[28] D. Wagner. Cryptanalysis of an algebraic privacy homomorphism. In *Proceedings of Information Security Conference ISC*, pages 234–239, Bristol, UK, Oct 2003. ISBN 3-540-20176-9.

[29] 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*, 5(10):1417–1431, Oct 2006. ISSN: 1536-1233.

[30] 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*, pages 356–367, Florence, Italy, 2006. ISBN 1-59593-368-9.

[31] 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, 2004. ISBN 0-7695-2136-3.

APPENDIX

Table 1: Simulation parameters

Parameter	Value
message length, payload	56 Bytes, 29 Bytes [14]
MAC, radio range	802.11 CSMA/CA, 10m
packet loss model	SNR-BOUNDED, -104 dBm noise
data rate	38400 Bit/s [14]
radio energy, CPU energy	16 mA [14], 5 mA [2,5]
per message energy	245 μAs [5]
cipher	RC5, 64 Bit key-/block-length [27]
energy per en-/decryption	1,3 μAs [2,20]
#SEEDs per sampling point	50, $< 2\%$ relative std. deviation