

## Article

# Symmetric-Key-Based Authentication among the Nodes in a Wireless Sensor and Actuator Network

Thibaut Vandervelden<sup>1,\*</sup>, Ruben De Smet<sup>2</sup>, Kris Steenhaut<sup>2</sup> and An Braeken<sup>1</sup>

<sup>1</sup> Department of Engineering Sciences and Technology (INDI), Vrije Universiteit Brussel (VUB), 1050 Brussels, Belgium; an.braeken@vub.be

<sup>2</sup> Department of Electronics and Informatics (ETRO), Vrije Universiteit Brussel (VUB), 1050 Brussels, Belgium; rdesmeta@etrovub.be (R.D.S.); ksteenha@etrovub.be (K.S.)

\* Correspondence: thvdveld@vub.be

**Abstract:** To enable today's industrial automation, a significant number of sensors and actuators are required. In order to obtain trust and isolate faults in the data collected by this network, protection against authenticity fraud and nonrepudiation is essential. In this paper, we propose a very efficient symmetric-key-based security mechanism to establish authentication and nonrepudiation among all the nodes including the gateway in a distributed cooperative network, without communicating additional security parameters to establish different types of session keys. The solution also offers confidentiality and anonymity in case there are no malicious nodes. If at most one of the nodes is compromised, authentication and nonrepudiation still remain valid. Even if more nodes get compromised, the impact is limited. Therefore, the proposed method drastically differs from the classical group key management schemes, where one compromised node completely breaks the system. The proposed method is mainly based on a hash chain with multiple outputs defined at the gateway and shared with the other nodes in the network.

**Keywords:** wireless sensor networks; authentication; nonrepudiation; cooperative network; anonymity; TESLA



**Citation:** Vandervelden, T.; De Smet, R.; Steenhaut, K.; Braeken, A. Symmetric-Key-Based Authentication among the Nodes in a Wireless Sensor and Actuator Network. *Sensors* **2022**, *22*, 1403. <https://doi.org/10.3390/s22041403>

Academic Editor: Giovanni Pau

Received: 31 December 2021

Accepted: 9 February 2022

Published: 11 February 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Wireless Sensor and Actuator Networks (WSANs) are currently very popular and are nowadays applied in a multitude of domains for monitoring and control. Application domains are, for instance, industrial automation, home automation, agriculture, transportation and logistics, body area networks, military applications, underground sensor networks, traffic monitoring, and air pollution monitoring. WSANs consist of a group of sensors, measuring different types of environmental parameters such as temperature, sound, humidity, and motion together with actuators equipped with, e.g., servos and motors that interact with them.

A very important requirement in WSANs is to present trustworthy measurements to the end users such that they can have confidence and accept conclusions drawn from them. At the same time, a long lifetime should be guaranteed with minimum to no maintenance overhead as the sensors are mostly battery powered and maintenance is very expensive. Therefore, adequate and efficient security mechanisms should be included, which are in the first place focused on the establishment of authentication and integrity.

Since efficiency is of utmost importance, symmetric-key-based solutions should be addressed. A very popular and elegant symmetric-key-based broadcast authentication mechanism for WSANs, called TESLA [1], was developed in 2002. However, this protocol only allows authentication by the root, also known as sink or cluster head, to the other nodes of the network. An effective symmetric-key-based mechanism, which also addresses authenticity, coming from the different nodes of a distributed network (without predefined positions) has not yet been proposed in literature as far as the authors are aware. This

shortcoming has been circumvented till now by the use of a common group key shared among all nodes in the network. It is not the most optimal and secure solution as it does not provide protection against malicious insiders, who can take the identity of the others without them being notified, and thus only offers authentication at the group level. Moreover, as soon as one of the nodes is captured and corrupted, the whole security of the network is broken.

## 2. Related Work

There exists a wide variety of solutions addressing key management in wireless sensor networks for different types of keys (pairwise keys, group keys, individual keys) for different types of network topologies (hierarchical, flat). Both symmetric- and asymmetric-key-based approaches have been proposed. For efficiency reasons, we focus on the most efficient ones—those being the symmetric-key-based ones. We also limit our discussion to key management schemes for WSNs, which address security at the group level, and exclude schemes such as SPINS [2] and BROS [3] whose focus is on the construction of pairwise keys.

One of the most complete symmetric key management schemes for WSNs is the Localized Encryption and Authentication Protocol (LEAP) [4]. This protocol for hierarchical networks explains how to establish a pairwise key between two cluster heads, a group key in the same cluster, and a network key.

Other symmetric-key-based group key management schemes for tree- and star-based networks can be found in [5–7]. In [5], the group key is constructed by means of the evaluation of a bivariate polynomial in combination with Lagrange interpolation. In [6,7], the group key is constructed using a generator matrix.

For distributed ad-hoc networks with randomly deployed nodes, there are only a limited number of symmetric-key-based proposals for key management since it is more difficult as the positions of the nodes in the network are not known in advance. In [8], the keys are generated by relying on a robot-assisted network bootstrapping technique. Recently, in [9], another approach has been proposed to derive multicast keys among a dynamic group of nodes by using a cloud-based Network Multicast Manager (NMM) and a dedicated asymmetric distribution of key material between NMM and a sensor node.

However, in all of these approaches, the focus is on the generation of a group key, sometimes combined with a pairwise key between two nodes or an individual key with the cluster node/sink. The authentication of the individual sensor nodes verifiable by each of the other members in the group has not yet been addressed via a cryptographic approach. If it is considered at all, trust- and reputation-based schemes are defined [10], which typically require very high computational efforts and are mainly installed at the gateway level.

Despite the needed efforts, the feature is worth considering, particularly for cooperative WSNs [11]. First, it allows the node to verify the trustworthiness of its neighbor nodes such that the data received from them can be used for own purposes (e.g., calibration). Second, it also enables to localize malicious nodes in the network and to avoid all types of corresponding attacks such as wormholes, sinkhole, Sybil, sleep, selective forward, and denial of service. For the detection of malicious nodes, several statistical [12] and AI-based methods [13] have been explained in literature. However, a solution based on a cryptographic approach is not yet available in literature, although it is one of the most effective assets to avoid these type of attacks.

For the authentication of the cluster node using symmetric key cryptography, the TESLA protocol [1] and its variants [14–16] offer currently the most efficient solution. These schemes are based on a hash chain and on an orchestrated release of each of the values of the chain in predefined timeslots. As a consequence, there is a delay of one timeslot to verify the authenticity of the message. Note that there is also the scheme of [17], which also uses a hash chain, but now combined with authenticated encryption resulting in a

scheme where there is no delay for authentication. The proposed scheme was applied in an edge-based architecture.

In this paper, we extend the usage of the TESLA protocol to all the sensor nodes in the network by relying on the extendable output feature of the sponge functions, proposed by Bertoni et al. in [18].

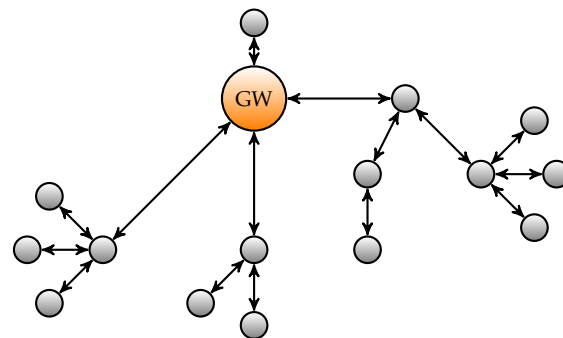
### 3. Preliminaries

First, we provide some background information on the network model and the corresponding attacker model. Next, the required security features are elaborated and a short description of the cryptographic operations used in our system is given.

#### 3.1. Network Model

We consider a distributed WSN, consisting of constrained sensors with limited energy and computational resources. The sensors can change position in the network (e.g., animal monitoring) or can stay at fixed positions (e.g., pollution monitoring at different predefined locations). We assume that changes in the network are limited. Nodes can be added, but if nodes are removed, then the memory on it should also be erased.

The gateway sends a broadcast message to all the sensor nodes in the network, which can reply by sending a message back to the gateway. Both communications can be done via a multihop approach. Each of the nodes on the routing path should be able to derive the content and origin of the message and verify its authenticity and integrity. A representation of such a network is depicted in Figure 1.



**Figure 1.** Representation of a WSN. The orange node represents the gateway of the network. The other nodes represent sensors or actuators in the network.

#### 3.2. Attacker Model

We make the following assumptions regarding the attack model.

- The attacker is active during the whole process and is able to eavesdrop the messages sent over the wireless channel. The attacker can actively change, delete, alter, and replay messages (or parts of it) sent over the channel.
- The gateway is considered trusted and possesses the required protection measures to resist security attacks. As a minimum, secure storage of key material is guaranteed.
- The security material in the nodes is stored in tamper-proof memory, such that it is difficult for an attacker to reveal it.

#### 3.3. Security Features

The classical security features of confidentiality, integrity, authentication, and anonymity together with nonrepudiation are attained at group level. This means that all nodes in the network are able to send messages satisfying these features as long as no network node is malicious. Compared with other schemes, we make it possible to address these features among all nodes, without storage of a shared pairwise key upfront or submission of random values to derive session keys during the actual operation.

The strong point in our scheme is that the security features of authentication, integrity, and nonrepudiation also apply on an individual level and are maintained as long as at most one of the nodes gets compromised. In schemes based on group keys, a malicious node can take the identity of another node of the group, which is not possible in our scheme. Our scheme protects against impersonation and man-in-the-middle attacks as long as at most one node is hijacked.

In case more than one node is compromised, the authentication and integrity of the gateway is still valid but they are impacted for a limited number of nodes, which vary in the different timeslots. Therefore, it also enables fast fault isolation and effective intrusion detection and avoidance.

### 3.4. Cryptographic Operations

Taking into account the limited computational resources and the minimum lifetime of three years to be guaranteed by the sensors, the overhead created due to ensuring the required security features should be as low as possible. Therefore, the cryptographic operations used in the proposed scheme are limited to xor operations, encryptions, and hash operations.

We denote the xor operation by  $\oplus$ , the symmetric key encryption by means of a common shared key  $K$  by  $E_K()$  and corresponding decryption by  $D_K()$ . We assume the usage of an Authenticated Encryption (AE) mode such as AES-GCM or AES-CCM.

The hash operation used in our scheme is based on the SHA3 standard SHAKE256( $M, d$ ), which is an algorithm providing an output with variable length  $d$  and providing a minimum security strength of  $\min(d/2, 256)$ -bit against collision attacks and  $\min(d, 256)$ -bit against preimage and second preimage attacks. We denote by  $H_i(M) = (M_1, M_2, \dots, M_i)$  the SHAKE256( $M, d$ ) algorithm working on message  $M$  providing  $i$  output blocks, where the total length of all blocks in the output vector equals to  $d$ ,  $|M_1| + |M_2| + \dots + |M_i| = d$ .

## 4. Proposed Scheme

The scheme consists of three different entities: nodes, gateway, and application server. We consider two different phases: key initialization and actual communication and authentication.

### 4.1. Key Initialization

Suppose there are  $2^n - 1$  nodes with identities  $ID_i = i \in \{1, \dots, 2^n - 1\}$  in the WSN. The gateway chooses two random keys  $K^0$  and  $K_e^0$  and computes a hash chain of both with length  $t$  and a corresponding output consisting of  $n + 1$  blocks and 1 block using the hash functions  $H_{n+1}$  and  $H_1$ , respectively. The hash chain of key  $K^0$  will be mainly used to derive material for authentication purposes and the hash chain of key  $K_e^0$  will define the encryption keys in the process. There are  $n$  independent components needed in the hash chain of  $K^0$  in order to be able to define unique authentication material for each of the  $2^n - 1$  nodes in the network.

$$\begin{aligned}
 j = 1 & : K^1 = H_{n+1}(K^0) = (\mathbf{K}_0^1, K_1^1, K_2^1, \dots, K_n^1), K_e^1 = H_1(K_e^0) \\
 j = 2 & : K^2 = H_{n+1}(\mathbf{K}_0^1) = (\mathbf{K}_0^2, K_1^2, K_2^2, \dots, K_n^2), K_e^2 = H_1(K_e^1) \\
 j = 3 & : K^3 = H_{n+1}(\mathbf{K}_0^2) = (\mathbf{K}_0^3, K_1^3, K_2^3, \dots, K_n^3), K_e^3 = H_1(K_e^2) \\
 & \vdots \\
 j = t & : K^t = H_{n+1}(\mathbf{K}_0^{t-1}) = (\mathbf{K}_0^t, K_1^t, K_2^t, \dots, K_n^t), K_e^t = H_1(K_e^{t-1})
 \end{aligned}$$

Note that for all  $j \in \{1, \dots, t\}$ , the variables  $K_0^j, K_e^j$  consist of 256 bits and  $K_s^j$  of 128 bits for all  $s \in \{1, \dots, n\}$ . We also denote the first 128 bits of the variables  $K_0^j$  and  $K_e^j$  by  $\bar{K}_0^j$  and  $\bar{K}_e^j$ .

Each node  $i$  receives  $t + 1, K_0^{t+1} = H_1(\mathbf{K}_0^t)$  installed, corresponding with the root of the first hash chain.

Define for each node  $i$  and timeslot  $j \in \{1, \dots, t\}$  the parameter  $b$ , which equals to

$$\begin{aligned} b &= (i + j \pmod{2^n - 1}) + 1 && \text{if } i + j \neq 2^n - 1 \\ &= 1 && \text{if } i + j = 2^n - 1 \end{aligned}$$

Denote the binary representation of  $b$  by  $b = (b_1, \dots, b_n)$ . Then, the node  $i$  also obtains for each timeslot  $j \in \{1, \dots, t\}$  the following tuple  $K_{ij} = (K_i^{j1}, K_i^{j2})$  installed:

$$\begin{aligned} K_i^{j1} &= \bigoplus_{s=1}^n b_s K_s^j \\ K_i^{j2} &= E_{\bar{K}_0^{j+1} \oplus K_i^{j1}}(\bar{K}_e^j) \end{aligned}$$

Note that the first variable in the tuple  $K_i^{j1}$  corresponds with a linear combination, defined by  $b$ , of the hash outputs of  $K_0^{j-1}$ . Since  $b$  varies for each node  $i$  and timestamp  $j$ , the overall complexity of the linear combinations can be considered similar for all nodes. The second variable  $K_i^{j2}$  enables deriving the first part of the hash chain of  $K_e^0$ .

The key  $K_i^{j1}$  is used in the later protocol to firstly verify the authentication of the hash chain of the gateway and to secondly authenticate its own message, which can be publicly verified by the other nodes in the next timestamp. The key  $K_i^{j2}$  enables the derivation of the decryption key of the encrypted message coming from the gateway, which is also used as the group encryption key by the nodes in the same timestamp. Note that this key  $\bar{K}_e^j$  can only be defined after reception of the value  $K_0^{j+1}$ , as this is required to decrypt  $K_i^{j2}$ .

The gateway installs the hash chain  $K^0, K_0^1, \dots, K_0^{t+1}$  and  $K_e^0, K_e^1, \dots, K_e^t$ . We consider nodes and gateway synchronized, which is also needed for several well-known medium access and radio duty cycling (RDC) protocols such as IEEE 802.15.4e Time Slotted Channel Hopping (TSCH).

**Example 1.** The construction of the key material for  $n = 2, t = 4$  and node  $ID_1$  ( $i = 1$ ) is as follows.

Construction of key material, starting from two root keys  $K^0$  and  $K_e^0$ .

$$\begin{aligned} j = 1 & : K^1 = H_3(K^0) = (K_0^1, K_1^1, K_2^1), K_e^1 = H_1(K_e^0) \\ j = 2 & : K^2 = H_3(K_0^1) = (K_0^2, K_1^2, K_2^2), K_e^2 = H_1(K_e^1) \\ j = 3 & : K^3 = H_3(K_0^2) = (K_0^3, K_1^3, K_2^3), K_e^3 = H_1(K_e^2) \\ j = 4 & : K^4 = H_3(K_0^3) = (K_0^4, K_1^4, K_2^4), K_e^4 = H_1(K_e^3) \end{aligned}$$

For  $ID_1$ , the following material is stored.

$$\begin{aligned} j = 1(b=3) & : K_1^{11} = K_1^1 \oplus K_2^1, K_1^{12} = E_{\bar{K}_0^2 \oplus K_1^{11}}(\bar{K}_e^1) \\ j = 2(b=1) & : K_1^{21} = K_1^2, K_1^{22} = E_{\bar{K}_0^3 \oplus K_1^{21}}(\bar{K}_e^2) \\ j = 3(b=2) & : K_1^{31} = K_2^3, K_1^{32} = E_{\bar{K}_0^4 \oplus K_1^{31}}(\bar{K}_e^3) \\ j = 4(b=3) & : K_1^{41} = K_1^4 \oplus K_2^4, K_1^{42} = E_{\bar{K}_0^5 \oplus K_1^{41}}(\bar{K}_e^4) \\ j = 5 & : K_0^5 = H_1(K_0^4) \end{aligned}$$

#### 4.2. Actual Communication and Authentication Process

In each timeslot  $j$ , the gateway starts with a broadcast message and the nodes react to this message. Both steps are explained in more detail below.

#### 4.2.1. Broadcast Message of Gateway

Denote by  $M^j$  the data to be broadcasted by the gateway at timeslot  $j \in \{t-1, \dots, 1\}$  with decreasing timeslots. The message  $M^j$  can include different types of information. For instance, it can contain the list of nodes from whom data are (not) received, the list of malicious nodes, and data on the clock to enable synchronization. For each timeslot  $j \in \{t-1, \dots, 1\}$ , the following information is broadcasted:

$$j, K_0^{j+1}, E_{\bar{K}_e^j}(M^j, H_1(M^j, K_0^j))$$

#### 4.2.2. Reaction of Nodes

Upon arrival of this message, the sensors in the network perform the following actions.

**Step 1:** First, the nodes are able to verify the authenticity of the gateway, but not the real authenticity of the received broadcast message of the gateway.

In order to do this, the authenticity of the received value  $K_0^{j+1}$  is verified by computing the hash of it. In case  $j = t-1$ , this corresponds with the prestored value. If  $j < t-1$ , comparison is made with the previously sent value and the prestored key material of slot  $j$ .

**Step 2:** Next, by construction of the key material stored in each node and the received value  $K_0^{j+1}$ , the node is able to extract  $\bar{K}_e^j = D_{\bar{K}_0^{j+1} \oplus K_i^{j1}}(K_i^{j2})$  and decrypt the last part of the broadcast message.

This key is also used to send, in the timeslot  $j$ , their own data  $M_i^j$  in the same way, by encoding it in the following format:

$$j, E_{\bar{K}_e^j}(i, M_i^j, H_1(M_i^j, K_i^{(j+1)1}))$$

Note that using the same key material, multiple messages can be sent by the nodes during this timeslot. An enumeration of the order can be included in the message. The number of messages that can be received in each slot depends on the receivers' buffer because messages are tentatively accepted based on the validation of the first part of the message and completely accepted on the next slot if the authentication tag is correct.

**Step 3:** In the same timeslot, the node is able to extract the received message  $M_r^j$  of all of its children or its neighbors (depending on the routing protocol) with identity  $r$  using the same decryption key  $\bar{K}_e^j$ . Again, this is an authenticity check at the group level. To enable it at the node level, the messages need to be stored such that they can be verified in the next timeslot.

The sensors also store the broadcast message  $(j, M^j, H_1(M^j, K_0^j), K_0^{j+1})$  for further verification in the next timeslot.

**Step 4:** Finally, the nodes can start verifying the individual authenticity of the messages stored from the previous timeslot  $j+1$  coming from the gateway and the children or neighboring nodes using the received value  $K_0^{j+1}$ .

In timeslot  $j$ , the gateway is also collecting all the messages from the nodes and is able to check the validity of the messages at the node level as the gateway possesses the required security material.

**Example 2.** The actual communication and authentication process for node  $ID_1$  in the case  $n = 2$  at timeslot  $j = 2$  consists of the following steps, based on the key material as described in Example 1.

- Step 0 : Gateway sends:  $2, K_0^3, E_{\bar{K}_e^2}(M^2, H_1(M^2, K_0^2))$
- Step 1 : Node:  $H_3(K_0^3) = (K_0^4, K_1^4, K_2^4)$ , and verifies  $K_0^4, K_1^{41}$
- Step 2 : Node:  $\bar{K}_e^2 = D_{\bar{K}_0^3 \oplus K_1^{21}}(K_1^{22})$ ,  
 $D_{\bar{K}_e^2}(M^2, H_1(M^2, K_0^2)), E_{\bar{K}_e^2}(2, M_1^2, H_1(M_1^2, K_1^{31}))$
- Step 3 : Node decrypts:  $(E_{\bar{K}_e^2}(2, M_r^2, H_1(M_r^2, K_r^{31})))_r$   
 : Node stores:  $\{2; (M^2, H_1(M^2, K_0^2)), (M_r^2, H_1(M_r^2, K_r^{31}))_r\}$
- Step 4 : Node checks:  $\{3; M^3, H_1(M^3, K_0^3), (M_r^3, H_1(M_r^3, K_r^{41}))_r\}$

#### 4.2.3. Addition/Removal of Nodes

Corresponding with the hash function  $H_{n+1}$ , there is security material for  $2^n - 1$  nodes. As long as there are less than  $2^n - 1$  nodes in the field, additional nodes can still securely join the network by storing the security material in the node.

In case a node needs to be removed, the memory on it should be completely erased as a malicious insider would be able to break the confidentiality and the anonymity of the consecutive communication (cf. Theorem 2).

We have assumed in the attacker model that the keys are stored in tamper-proof memory; therefore, an attacker will not be able to simply extract the key material of the node. The problem occurs in the case of so-called insider attacks, where nodes become malicious. In this situation, the node's identity should be noted. If more potential malicious nodes pop up, a list of untrusted inputs should be made, based on the identities of the malicious nodes (cf. Theorem 3).

### 5. Security Evaluation

For the security proofs, we consider the usage of building blocks that are semantically secure under chosen plaintext attacks. As a consequence, the generated ciphertext and authentication tag can be considered indistinguishable from the ones generated by a random permutation or random function, respectively. To be more specific, in this proposed scheme, the selected algorithms  $\text{SHAKE256}(M, d)$  and AES satisfy this feature.

**Theorem 1.** The protocol is able to offer confidentiality, integrity, authentication, anonymity, and nonrepudiation with a minimum security level of 128 bits with the condition that there are no malicious nodes in the network and the underlying security algorithms are not broken.

**Proof.** First, we prove the statement from the gateway point of view, where the focus is on authentication, integrity, and confidentiality. A valid key  $K_0^{j+1}$  in the broadcast message cannot be constructed by an attacker if the underlying hash algorithm offers protection against collision and preimage attacks, which has, in the best situation, a probability of  $2^{-128}$  in the case of the  $\text{SHAKE256}(M, d)$  protocol. Without knowledge of the value  $\bar{K}_e^j$ , which is impossible to derive without having additional information (such as the trusted nodes) the broadcast message cannot be decrypted and is thus protected against confidentiality and integrity with a probability of  $2^{-128}$ , corresponding with a brute force guess of the key. In addition, due to the security of the building blocks under chosen plaintext attacks, the generated ciphertexts cannot be differentiated from a random value.

For the trusted nodes, the transmitted message cannot be decrypted by attackers eavesdropping on the channel as they are encrypted with the same key used by the gateway; thus, they have a probability of  $2^{-128}$  to be successful. In addition, the message cannot be distinguished from a randomly generated value due to the strength of the encryption algorithm with respect to chosen plaintext attacks. As the message cannot be decrypted,

the identity remains hidden and, due to the usage of an authenticated encryption mode, the integrity is also provided. The authentication is offered in the next timeslot as the hash in the encrypted message can then be validated. Only the node with the correct prestored information is able to derive this hash.  $\square$

**Theorem 2.** *The protocol is able to offer integrity, authentication, and nonrepudiation with a minimum security level of 128 bits in case there is at most one malicious node in the network.*

**Proof.** If there is one malicious node in the network, it is able to derive the encryption key used to encrypt the messages of the nodes and the gateway and, thus, break the confidentiality and the anonymity of the messages. However, this node possesses no further information to break the authenticity of the messages from either the gateway or the other nodes, since it requires knowledge of prestored information to derive  $K_i^{j1}$ , which is only available in the other nodes and gateway due to the unique construction of  $b$  on which  $K_i^{j1}$  is computed. Due to the strength of the hash function with respect to preimage attacks, it is impossible to find a collision on the hash function.  $\square$

**Theorem 3.** *The protocol is not able to offer integrity, authentication, and nonrepudiation with a minimum security level of 128 bits in at most  $\frac{2^m}{2^n-1}\%$  of nodes per timeslot in the case where there are  $m$  malicious nodes collaborating in the network.*

**Proof.** If there are  $m$  malicious nodes collaborating in the network, they are able to construct the security material of all nodes with identity  $i$  for which  $b = (i + j \bmod (2^n - 1)) + 1$  can be constructed as a linear combination of the values  $m_i + j \bmod (2^n - 1) + 1$ , with  $m_i$  being the identities of the malicious nodes. In total, for each timeslot, there are at most  $2^m - m - 1$  other nodes besides the malicious nodes involved. Once the security material is revealed, the security features of these nodes are not addressed anymore.

However, note that due to the construction of the key material, which is divided in a circular way, the infected  $2^m - m - 1$  nodes will vary over the different and independent timeslots. Due to the nonrepudiation feature, the source of the attack can be easily retrieved.  $\square$

With respect to the resistance against Denial of Service (DoS) attacks, we notice that it is resistant as long as the nodes are not malicious. Following the attacker model (second point), the gateway is considered to be trusted, so it is impossible that  $K_0^{j+1}$  is leaked by assumption. It is also impossible to derive another value of  $K_0^{j+1}$  for an outsider, which might be valid (because of Theorem 1). Moreover, there are two checks that need to be done, the one on  $K_0^{j+1}$  and  $K_1^{(j+1),1}$ . Even in the extremely unlikely case the attacker knows the value  $K_0^{j+1}$ , it should also know the exact value of  $\bar{K}_e^j$  since the nodes can already do the decryption; if it is useless, it may assume that the message is invalid. If  $\bar{K}_e^j$  is leaked, which is due to the strength of the encryption algorithm only possible by means of exhaustive search, a DoS attack becomes possible. Therefore, malicious insider nodes can still overload the network, but this is a problem that is very difficult to overcome.

## 6. Performance Evaluation

### 6.1. Computational Cost

We focus on the side of the nodes in the network and will determine the computational efforts for each of the four steps to be performed as the reaction of the nodes on reception of the broadcast message of the gateway (cf. Section 4.2.2). We concentrate here on the most costly cryptographic operations, these being encryption and hash. The cost for each of the four steps is equal to the following:

**Step 1:** 1 hash operation with input 256 bits and output length  $128n + 256$  bits.



**Step 2:** 1 decryption operation on 128 bit message, 1 decryption operation on a message with length  $256 + |M^j|$  bits, and 1 encryption on a message with bit length  $n + 256 + |M_i^j|$ .

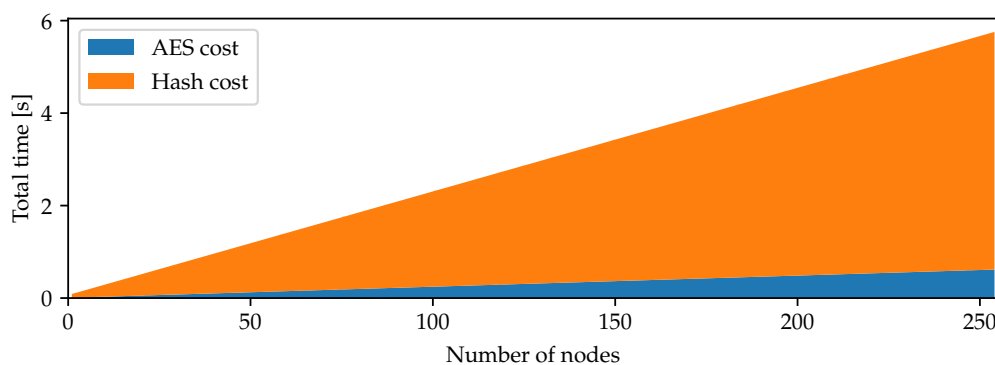
**Step 3:**  $n_r$  decryptions of messages with bit length  $n + 256 + |M_i^j|$ , where  $n_r$  is the number of nodes from whom the node wants to derive the message (neighbors, children in routing scheme) and verify the authenticity.

**Step 4:** 1 hash operation on a message with length  $256 + |M_i^j|$ , output 256 bits;  $n_r$  hash operations on messages with input a message with length  $256 + |M_i^j|$  bits, output 256 bits.

As a proof of concept, we implemented our protocol on a constrained device, the Zolertia RE-Mote development board, which contains a CC2538 microcontroller using the ARM Cortex-M3 architecture. The device possesses 512 kB flash memory and has 32 kB RAM.

The reason why we only used this board for our measurements is because the behavior of the cryptographic primitives used is the same for the different ARM Cortex-M architectures, as shown for example in [19]. Timing measurements are made using the Data Watchpoint and Trace (DWT) unit of the microcontroller. This unit contains a monotonically increasing counter, which increases at the same frequency as the core of the microcontroller. The error on the measurements is, thus, negligible.

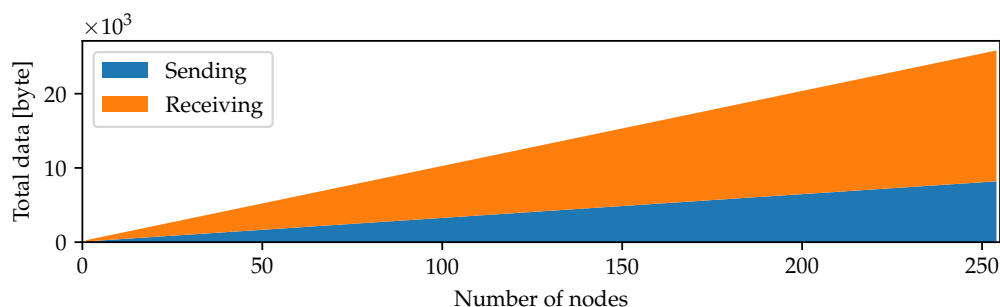
Figure 2 shows the time it takes for an individual node to perform the required security actions in case it has to check  $n_r$  neighboring nodes. As a consequence, we can conclude that it takes less than 0.18 s if every node has less than 5 neighboring nodes to verify, while it takes less than 6 s if every node has to check the input of 250 neighboring nodes. As a consequence, this performance shows that the protocol is feasible for realistic settings.



**Figure 2.** Time required for a Zolertia RE-Mote to perform the required security actions as a function of the number of neighboring nodes.

## 6.2. Communication Cost

Denote the average length of the message  $M^j, M_i^j$  by  $|M|$ . During each timeslot, the node receives a message of length  $(n_r + 1) \log_2 t + (2 + n_r) \cdot 256 + n_r \cdot n + (n_r + 1)|M|$ . The node sends, during each timeslot, its own message of length  $\log_2 t + n + |M| + 256$  and forwards the message of its  $n_r$  children of the same length. There are no additional intermediate messages needed in the protocol to define separate session keys. Figure 3 shows the total number of sent and received bytes in case the length of the transmitted message of all nodes equals 32 bytes for a varying number of neighboring nodes. If a node has less than 5 neighboring nodes, 229 bytes will be sent and 445 bytes received over the channel. This demonstrates again that the overhead is reasonable for realistic scenarios.



**Figure 3.** Amount of data sent and received during communication as a function of the number of neighboring nodes. Here, we assume the length of the transmitted messages to be equal to 32 bytes.

### 6.3. Storage Cost

The storage cost is an inherent issue of the TESLA protocol. In order to still show the feasibility of our approach, we now provide a concrete example. The storage cost on each node mainly depends on the length of the hash chain and, thus, also the number of timeslots  $t$  in which authenticated messages can be sent. Since there is no relation between the individual values, there is no smart way to optimize the storage as can be typically done for a hash chain. Therefore, the storage size is dominated by  $8t$  bytes. Considering a situation of 12 timeslots a day, corresponding with one every 2 h, the storage area requires 384 bytes per day or 140 kB per year, and thus, 420 kB for 3 years, which is feasible on a Zolertia RE-Mote given the fact that it contains 512 kB of flash memory.

## 7. Conclusions and Future Work

This paper proposes an effective and highly efficient scheme to enable broadcast authentication in a distributed wireless sensor network containing constrained devices. In particular, it represents the first symmetric-key-based solution to address authentication and nonrepudiation at the node level without separate key agreement sessions among the communicating nodes. Moreover, besides authentication and nonrepudiation, the proposed protocol is also able to offer confidentiality and anonymity without the presence of malicious nodes. Even with at most one malicious node, authentication and nonrepudiation still remain valid. The impact is also limited when there are more malicious nodes collaborating.

The proposed scheme is constructed by means of a multivector hash chain shared among the different nodes in the network. The scheme perfectly fits for applications enabling low-cost sensing and actuating, where trust and fault isolation are essential.

Due to the underlying construction of a hash chain relying on timeslotted communication, the scheme also possesses some inherent issues. First, while confidentiality and group-based authentication are obtained in the same timeslot, the actual node-based authentication can only be performed in the next timeslot. However, we believe that this is an acceptable trade-off between security and efficiency. Second, as there is no inherent structure in the key chains to be stored at the different nodes, the storage cost is proportional to the number of timeslots to which security can be offered. However, we have shown that the security lifetime for a popular lightweight node is still realistic.

Finally, we need to make sure that the memory of the nodes leaving the network is erased in order to avoid protection against collaboration of inside malicious nodes; thus, applications relying on limited changes in the network topology are more suitable.

In the future, from a theoretical point of view, we will investigate how patterns among malicious nodes evolve through the different noninfected nodes over the different timeslots. From a practical point of view, we will set-up a real application—e.g., for CO<sub>2</sub> monitoring in a building—and study the impact of different network parameters on the efficiency of the scheme.

**Author Contributions:** Conceptualization, A.B.; methodology, A.B.; software, T.V.; validation, T.V. and R.D.S.; formal analysis, A.B. and K.S.; investigation, A.B., T.V., R.D.S.; resources, K.S.; data curation, T.V.; writing—original draft preparation, A.B.; writing—review and editing, T.V., R.D.S., and K.S.; visualization, T.V.; supervision, K.S.; project administration, A.B.; funding acquisition, K.S. and A.B. All authors have read and agreed to the published version of the manuscript.

**Funding:** The research was partly supported by the project fund for technology and transfer (Tetra) of Flanders Innovation & Entrepreneurship (Vlaio) under grant number HBC.2020.2073–Velcro.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Perrig, A.; Canetti, R.; Tygar, J.; Song, D. The TESLA broadcast authentication protocol. *RSA CryptoBytes* **2002**, *5*, 2–13.
2. Perrig, A.; Szewczyk, R.; Tygar, J.D.; Wen, V.; Culler, D.E. Spins: Security protocols for sensor networks. *Wireless Netw.* **2002**, *8*, 521–534. [[CrossRef](#)]
3. Lai, B.C.; Hwang, D.D.; Kim, S.P.; Verbauwhede, I. Reducing radio energy consumption of key management protocols for wireless sensor networks. In Proceedings of the ISLPED'04: 2004 International Symposium on Low Power Electronics and Design, Newport Beach, CA, USA, 9–11 August 2004; pp. 351–356.
4. Zhu, S.; Setia, S.; Jajodia, S. LEAP: Efficient security mechanisms for large-scale distributed sensor networks. *ACM Trans. Sens. Netw.* **2006**, *2*, 500–528. [[CrossRef](#)]
5. Zhang, Y.; Shen, Y.Y.; Lee, S. A Cluster-Based Group Key Management Scheme for Wireless Sensor Networks. In Proceedings of the 2010 12th International Asia-Pacific Web Conference, Busan, Korea, 6–8 April 2010; pp. 3–5.
6. Li, L.; Wang, X. A High Security Dynamic Secret Key Management Scheme for Wireless Sensor Networks. In Proceedings of the 2010 Third International Symposium on Intelligent Information Technology and Security Informatics, Jinggangshan, China, 2–4 April 2010; pp. 507–510.
7. Shnaikat, K.N.; Alqudah, A.A. Key Management Techniques in Wireless Sensor Networks. *Int. J. Netw. Secur. Appl.* **2014**, *6*, 49. [[CrossRef](#)]
8. Ren, K.; Lou, W.; Zhu, B.; Jajodia, S. Secure and efficient multicast in wireless sensor networks allowing ad hoc group formation. *IEEE Trans. Veh. Technol.* **2009**, *58*, 2018–2029. [[CrossRef](#)]
9. Carlier, M.; Steenhaut, K.; Braeken, A. Symmetric-Key-Based Security for Multicast Communication in Wireless Sensor Networks. *Computers* **2019**, *8*, 27. [[CrossRef](#)]
10. Ding, J.; Zhang, H.; Guo, Z.; Wu, Y. The DPC-Based Scheme for Detecting Selective Forwarding in Clustered Wireless Sensor Networks. *IEEE Access* **2021**, *9*, 20954–20967. [[CrossRef](#)]
11. Gautam, A.K.; Kumar, R. A comprehensive study on key management, authentication and trust management techniques in wireless sensor networks. *SN Appl. Sci.* **2021**, *3*, 50. [[CrossRef](#)]
12. JHo, W.; Wright, M.; Das, S.K. Distributed detection of mobile malicious node attacks in wireless sensor networks. *Ad Hoc Netw.* **2012**, *10*, 512–523.
13. Umba, S.M.W.; Abu-Mahfouz, A.M.; Raamotsoela, T.D.; Hanck, G.P. Review of artificial intelligence based intrusion detection for software-defined wireless sensor networks. In Proceedings of the IEEE 28th International Symposium on Industrial Electronics (ISIE), Vancouver, BC, Canada, 12–14 June 2019.
14. Liu, D.; Ning, P. Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks. In Proceedings of the NDSS, San Diego, CA, USA, 6 February 2003; pp. 263–276.
15. Liu, D.; Ning, P. Multi-level  $\mu$ TESLA: Broadcast authentication for distributed sensor networks. *ACM Trans. Embed. Comput. Syst.* **2004**, *3*, 800–836. [[CrossRef](#)]
16. Liu, D.; Ning, P.; Zhu, S.; Jajodia, S. Practical broadcast authentication in sensor networks. In Proceedings of the MobiQuitous, San Diego, CA, USA, 17–21 July 2005; pp. 118–132.
17. Nakkar, M.; Altawy, R.; Youssef, A. Lightweight Broadcast Authentication Protocol for Edge-Based Applications. *IEEE Internet Things J.* **2020**, *7*, 11766–11777. [[CrossRef](#)]
18. Bertoni, G.; Daemen, J.; Peeters, M.; van Assche, G. Sponge-Based Pseudo-Random Number Generators. In *Cryptographic Hardware and Embedded Systems, CHES 2010*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2010; pp. 33–47.
19. Vandervelden, T.; de Smet, R.; Steenhaut, K.; Braeken, A. SHA3 and Keccak variants computation speeds on constrained devices. *Future Gener. Comput. Syst.* **2022**, *128*, 28–35. [[CrossRef](#)]