SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks (2015)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.1296

RESEARCH ARTICLE

Secure broadcast in distributed networks with strong adversaries

Pawel Szalachowski[1]* and Tiffany Hyun-Jin Kim[2]

[1] Department of Computer Science, ETH Zurich, Switzerland
[2] CyLab, Carnegie Mellon University, Pittsburgh, PA, U.S.A.

ABSTRACT

This paper proposes a framework that enables secure *one-to-many* communication for networks with limited capabilities in the face of a strong adversary that can capture an arbitrary set of nodes. Our approach consists of two main components: (a) group key establishment protocol and (b) special key management. Especially, we try to address the following question: *How strong of security properties can we achieve for broadcast communication in hardware-limited networks with a strong adversary?* We propose approaches and their variants that neither require special hardware nor use costly cryptographic operations. With thorough security and efficiency analyses, we discuss how our solutions can be applied to a variety of hardware-limited distributed systems. We also describe the implementation and evaluation results of the most promising variants. Copyright © 2015 John Wiley & Sons, Ltd.

KEYWORDS

network security; secure group communication; cryptographic protocols; strong adversary; distributed hardware-limited networks

***Correspondence**

Pawel Szalachowski, Department of Computer Science, ETH Zurich, CAB F 85.2, Universitatstrasse 6, 8092 Zurich, Switzerland.
E-mail: psz@inf.ethz.ch

1. INTRODUCTION

Achieving a high level of security in distributed and hardware-limited environments such as wireless sensor networks (WSN) is a well-known challenge [1,2], because of the following characteristics:

- The communication medium is open; consequently, it is easy to eavesdrop and modify traffic.
- Nodes are distributed and unattended. In many scenarios, network itself is used for monitoring.
- Nodes are cheap, without hardware protection, and their processing power is low.
- A strong adversary model is realistic, where an adversary can capture [3] an arbitrary set of nodes.

Successful solutions must be efficient for long-term deployment. Thus, one of the core design goals is to minimize the computation, storage, and transmission overhead on regular nodes. However, such a goal may be in conflict with the security properties that the solutions must provide.

The main contribution of this work is to propose and analyze a framework for secure broadcasting in the face of an adversary that can control the traffic and capture arbitrary network elements. In this framework, security is considered as secrecy of the broadcast communication. Our proposed solutions include the following high-level elements:

- efficient group key establishment method (Section 4.1); and
- mechanisms to protect secret values (keys) at nodes (Section 4.3).

The combination of these two elements results in methods providing different levels of security and performance guarantees, which we investigate throughout this paper.

Our approaches require neither special hardware modules nor costly public-key cryptography, and they are efficient even for long-term deployment. We analyze the proposed solutions as well as some of their variants and implement one that provides the best trade-off between security and efficiency. The proposed solutions can be especially useful in applications where nodes are unattended and prone to physical attacks. Example scenarios include command and control, area and industrial monitoring, and telemetry.

This paper is structured as follows. In Section 2, we postulate the problem by introducing the model of protected system and the required cryptographic primitives. We also define the attacker model and the desired security goals. In Section 3, we provide the high-level overview of the proposed approach, which is described in detail in Section 4. In Section 5, we present the security, efficiency, and usability analyses of the proposed method, and we present the modification and the implementation in Section 6. In Section 7, we conduct comparative efficiency analysis. Section 8 reviews related work, and Section 9 summarizes our findings.

2. PRELIMINARIES

2.1. System model

Distributed networks are composed of one special node called a *broadcast center* (BC) and regular nodes. The BC has sufficient hardware capabilities to perform efficient computations and to broadcast messages to all nodes. Unlike the BC, regular nodes are hardware-limited; that is, their computational, storage, and transmission capabilities are significantly lower than the capabilities of the BC. All regular nodes have a unique ID, which is a numeric value from 1 to n, and are capable of performing (some chosen) cryptographic operations and storing small secret values. We assume that the BC (or the network administrator) can pre-load some secret values to the regular nodes before their deployment.

In distributed networks, we consider the *one-way* and *one-to-many* communication models when the BC broadcasts messages to regular nodes. In terms of the trust model, we do not assume that nodes are physically protected and hence do not consider regular nodes as trusted entities. Although the BC is generally considered as the only trusted element in the system, we consider the scenario where an attacker captures and controls the BC.

2.2. Cryptographic primitives

We utilize symmetric-key (or private-key) cryptography, which is proven to be efficient in terms of computation, energy consumption, and storage overhead [4].

We also use a *pseudorandom keyed function* (PRF) $F : K \times X \rightarrow Y$, where K is the key space (key $k \in K$) and $F_k(\cdot)$ denotes its execution; one-way function is denoted as F_0. We highlight important properties of a PRF: (1) A function is a PRF if no adversary (which can be modeled using a polynomial-time algorithm) can distinguish whether it is interacting with F_k or some other function chosen at random from a set of all functions mapping X to Y; (2) a PRF is *non-invertible*, such that an adversary cannot determine x given $F_k(x)$; and (3) a PRF provides *key secrecy* such that interacting with F_k does not enable an attacker to compute key k.

We use a cryptographic primitive called *broadcast encryption* (BE), the goal of which is to encrypt (and broadcast) messages in such a way that only members of the *privileged* set can decrypt it. Members are called nodes or users that can be identified using their unique IDs. U denotes a set of all users, S represents a privileged set, and $R = U \setminus S$ is an *unprivileged* set, whose members cannot decrypt transmitted messages. Note that the membership of the privileged set can change dynamically.

We use the following algorithms to define BE:

- $Gen(n)$ takes the number of users $n = |U|$ and returns their initial private keys $k_1, \ldots, k_n$.
- For a set of keys of privileged nodes, $Enc(\{k_x | x \in S\})$ generates the following: (1) session key SK, which is a shared group key for privileged nodes for the session, and (2) header hdr, which helps the node derive SK with the corresponding priviledged node's private key. Note that hdr does not contain the private keys of privileged nodes.
- $Dec(k_{id}, S, hdr)$ takes a user's private key k_{id}, the set $S \subseteq \{1, \ldots, n\}$, and the hdr as inputs and returns session key SK if $id \in S$ or $\perp$ otherwise.

The aforementioned definition describes BE in a symmetric setting, where every user shares its private key with the BC. The BC executes $Gen(n)$ once and then pre-loads each of the generated keys into the corresponding node. In order to establish the group session key, the BC determines privileged set S, computes hdr, and broadcasts these values. Based on the received S and hdr, privileged nodes obtain session key SK, which is a group key only known to the members of S. A single transmission of hdr is called a *session* and every session is numbered uniquely in an ascending order from 1.

Efficiency of BE schemes is evaluated using the following measures: (1) *computational overhead*, or the required number of operations; (2) *storage overhead*, or the required number of stored keys; and (3) *transmission overhead*, or the size of the transmitted header. All measures are for both the BC and nodes. BE schemes are generally divided into *stateless* and *stateful*. Stateless schemes do not require a regular node to update its internal state in each session, while stateful schemes require such an operation.

2.3. Attacker model and security goals

The main goal of an attacker is to decrypt broadcast messages, which is equivalent to obtaining the session key SK. Conversely, the goal of the BC is to prepare and broadcast transmission in a way that only the members of the privileged set can decrypt it.

Because nodes are physically unprotected, an adversary can almost freely observe and modify traffic. Furthermore, an adversary has an access to various secret values (short- and long-term private keys), and in particular, we consider the following two cases:

DOI: 10.1002/sec

(1) When an adversary has the private keys of the unprivileged nodes only, his goal is to obtain the group key of the past or the current session.
(2) When an adversary has the private key(s) of the privileged node(s), his goal is to obtain the group key(s) of any previous session(s).

However, the proposed protocols must be secure in the presence of relaxed, eavesdropping adversaries. In order to securely broadcast messages, the system must support the session key establishment for a dynamically changing set of users. Consequently, the following security properties must be ensured. First, new members of privileged set S for a given session should not be able to decrypt any messages from the previous sessions (*backward secrecy (BS)*). Second, any revoked node (removed from set S) cannot decrypt the messages in the current and future sessions (*forward secrecy (FS)*). Third, the BE system must be *resilient* i.e., an adversary with even all the keys from set R of unprivileged nodes, still cannot compute a session key. This model is especially interesting in the case of payment-based multimedia broadcasting systems in which users without any subscription would collude to access the media for free. Next, the attacker able to capture arbitrary nodes, even from the set S, should not be able to decrypt the messages in the previous sessions (*break-backward protection (BBP)* or *perfect forward secrecy*). (Capturing a privileged node automatically enables the attacker to obtain the current and even the future session keys if it remains undetected.) Finally, the system should support *traitor tracing* to find a node that leaked its private key or decrypted content. Phan *et al.* describe security notions of BE and their relations in detail [5].

We consider detection of an adversary to be outside the scope of this paper. However, approaches presented throughout this paper can cooperate successfully with such mechanisms. For instance, in the case of a battlefield, the BC can observe terrain and decide which assets are already occupied by an adversary. Using our methods, if the BC can detect captured nodes, removing these nodes from the privileged set would prevent the adversary from decrypting any previous and following transmission. When captured nodes are undetected, an adversary would not be able to decrypt any previous session keys at least.

3. APPROACH

An efficient broadcast encryption scheme is necessary for our framework; unfortunately, it cannot be chosen arbitrarily. By examining the definition of the BBP property, it is trivial to notice that stateless BE schemes cannot provide BBP. Let us denote the following notations:

hdr^i : header in ith session,
S^i : privileged set in session i, consisting of the privileged nodes' IDs,
SK^i : session key in session i (i.e., group key of S^i's members), and
k_j^i : private key of jth user in ith session (shared with the BC).

Let us assume that user j is in the privileged set in sessions $i-1$ and i. Note that $k_j^{i-1} = k_j^i$, because users do not update their private keys in the stateless scheme. Thus, if an attacker successfully eavesdrops headers hdr^{i-1} and hdr^i and captures node j in session i, he or she can easily compute the previous session key: $SK^{i-1} = Dec(k_j^i, S^{i-1}, hdr^{i-1})$. Consequently, no stateless schemes can hold BBP, and proper key management techniques must be employed to ensure BBP. Such techniques must include nodes' private key update mechanism for every session.

We now outline an execution of the protocol. In the *initialization phase*, the BC executes $Gen(|U|)$ function, saves the resulting long-term private keys, and pre-loads each key into the corresponding node. Note that this phase is performed only once. After keys are pre-loaded, the actual protocol starts, and the BC performs the following for ith session:

(1) Set the session number to i and select privileged set S^i.
(2) Generate new, *session-specific* private key k_x^i for each member $x \in S^i$ for session i (various generation methods are presented in Section 4.3).
(3) Create session key and the header for session i using the members' session-specific private keys: $SK^i, hdr^i = Enc(\{k_x^i | x \in S^i\})$.
(4) Authenticate and broadcast i, S^i, and hdr^i.
(5) Broadcast traffic encrypted using SK^i.

Each node $x \in S^i$ performs the following actions:

(1) Receive i, S^i, and hdr^i.
(2) Exit if $x \notin S^i$ or authentication fails.
(3) Derive session-specific private key k_x^i for session i (we introduce the derivation methods in Section 4.3).
(4) Derive $SK^i = Dec\left(k_x^i, S^i, hdr^i\right)$.
(5) Receive and decrypt traffic using SK^i.

For each session, nodes in S_i and the BC establish the shared session key SK_i to protect messages in that session. Authentication and protection of the broadcast transmission are out of scope of this paper, as existing approaches can be applied [6,7,33]. If the BC wants to add or remove nodes from the privileged set, it executes the protocol again for the next session $i+1$.

4. DETAILS OF THE APPROACH

Two main elements are required to build the system as described in Section 3. First, a secure and efficient

Protocol 1: Realization of broadcast encryption [8].

$GF(q)$ - finite field with q elements (q is a large prime number),
interpolate(T) - interpolates polynomial based on points of T (can be implemented, e.g., by Lagrange interpolation),
$a \leftarrow B$ - element a is selected uniformly at random from set B.

Gen(n):
- $X = \{0\}$
- for $i \in \{1, \ldots, n\}$ do
  - $x_i \leftarrow GF(q) \setminus X$
  - $X = X \cup \{x_i\}$
  - $y_i \leftarrow GF(q)$
  - $k_i = (x_i, y_i)$
- return $k_1, \ldots, k_n$

Dec(k_{id}, S, hdr):
- if $id \notin S$ then
  - return $\perp$
- $K_{id} = (x_{id}, y_{id})$
- $T = hdr \cup \{K_{id}\}$
- $P = interpolate(T)$
- $SK = P(0)$
- return SK

Enc$(\{(x_i, y_i) | i \in S\})$:
- $SK \leftarrow GF(q)$
- $k = |S|$
- find polynomial P of degree k such that:
  - $SK = P(0)$
  - $\forall i \in S : y_i = P(x_i)$
  - $\forall i \in R : y_i \neq P(x_i)$
- $T = \{\}$
- for $i \in \{1, \ldots, k\}$ do
  - $(x, y) \leftarrow P \setminus (T \cup S \cup \{(0, SK)\})$
  - $T = T \cup \{(x, y)\}$
- $hdr = T$.
- return SK, hdr.

broadcast encryption scheme (*Gen*, *Enc*, and *Dec* functions) is needed to generate the session key for the privileged nodes. Next, for each session, a key management is required to update the session-specific private key between the BC and each node in S while providing the desired security properties (Section 2.3). In this section, we describe these two elements in detail.

4.1. Broadcast encryption

Berkovits introduced one of the first BE schemes [8] utilizing Shamir's scheme [9] for practical consideration. In Protocol 1, we introduce the concrete and optimized broadcast encryption scheme, which employs Shamir's scheme. In comparison with the original construction, our scheme omits special auxiliary secrets such that neither additional secrets need to be sent nor processed. Note that this optimization does not affect security.

A node's private key is represented as point $(x, y) \in GF(q)^2$, and the session key is represented as $(0, SK) \in GF(q)^2$. Algorithm *Enc* creates polynomial P, which goes through $(0, SK)$ and keys (points) for users in S, but P does not go through any of the keys (points) for users in R. Header hdr is constructed as a set of $|S|$ random points that belong to P. After receiving these points, each node executes function *interpolate*(), which interpolates polynomial P using points from hdr and the current node's private key. Finally, session key SK is computed as $P(0)$.

To establish a new session key, the BC needs to send $|S|$ messages, while each node stores only one private key.

Algorithm 2: One-time exploitation attack for Protocol 1 executed multiple times.

x, i, j satisfy: $i \neq j \wedge x \in S^i \cap S^j$,
$\mathcal{O}_1(x)$ - function which returns current private key k_x of xth node.

$\mathcal{A}(hdr^i, hdr^j, x)$:
- $k_x = \mathcal{O}_1(x)$
- $P^i = interpolate(hdr^i \cup \{k_x\})$
- $P^j = interpolate(hdr^j \cup \{k_x\})$
- return $\{(X, Y) : Y = P^i(X) \wedge Y = P^j(X)\}$

Note that the computation of the session key is performed using the polynomial interpolation, which can be executed fast, even on the low-end hardware (Section 6).

This scheme is resilient, and its security depends on the security of Shamir's protocol [8,9]. Unfortunately, when users' private keys are not updated, the presented solution is insecure as we illustrate next.

4.2. Attacks on BE scheme

Insecurity of the BE scheme is caused by its *one-time* property. Let us assume that the BE protocol is used multiple times (with the same long-term private keys of the users). Then the protocol can be attacked as presented in Algorithm 2. An attacker with only one node's private key can compute all private keys of users in $S^i \cap S^j$. Such a vulnerability makes this protocol unacceptable, and Schwenk discusses similar attacks on *one-time* schemes [10].

Besides the weakness against this easy-to-launch and effective attack, this scheme has other issues. For example, if node x was in the privileged set in some previous sessions, the attacker with only k_x can decrypt all the messages in the previous sessions.

Of course, the most powerful attack is capturing the node such that the adversary has access to not only all the secrets but also the *memory* of the node (i.e., the adversary knows all the past network traffic). In such a condition, it is impossible to protect the secrecy of the current session (and the next sessions if the attacker remains undetected), but a proper scheme can protect the confidentiality of the previous sessions (equivalent to BBP property). Hence, proper key managements are required to address the attacks as described above.

4.3. Key management

As a stateless scheme cannot hold BBP, a state for the sessions must be introduced, which requires some type of synchronization between the BC and regular nodes (BC and corresponding nodes must use the appropriate keys for each session). One way to achieve this is to send the unique session number and use it during the key update process. More specifically, the BC sets the session number, updates the privileged nodes' session-specific private keys, and uses them to create the header with the session

key. Each privileged node updates its own session-specific private key and uses it to obtain the session key.

The rest of this section introduces four key update methods. For simplicity, PRF is defined as $F : GF(q)^2 \times GF(q)^2 \rightarrow GF(q)^2$. This is due to the BE scheme, where keys are represented as points in $GF(q)^2$.

Method 1. This method generates the session-specific private key based on session number i and the long-term private key as follows:

$$k_j^i = F_{s_j}(i), \tag{1}$$

where s_j stands for a long-term private key that jth node shares with the BC. In each session, the BC performs a key update operation $k_x^i = F_{s_x}(i)$ for every privileged node $x \in S$. Each privileged node stores only one long-term private key and performs only one operation per session to derive the session-specific key. This key update scheme provides an unpredictable key for each session, and because of the PRF's properties, the long-term key remains secret, even if an attacker successfully obtains the session-specific private keys from multiple sessions. Moreover, this scheme protects the attack as presented in Algorithm 2.

Schwenk proposed a similar solution to mitigate the one-time property using a random value instead of the session number [10]. Unfortunately, Schwenk's approach does not provide BBP: Given the adversary that captures a node as presented in Algorithm 3, two consecutive sessions' headers and the long-term secret of a privileged node reveals the previous session keys.

Algorithm 3: Node capture attack for Method 1.

$\mathcal{O}_2(x)$ - a function that returns long-term secret key s_x of xth node,
i, j satisfy: $j \in S^i \cap S^{i-1}$.

$\mathcal{A}(hdr^{i-1}, hdr^i, S^{i-1}, S^i, j)$:
- $s_j = \mathcal{O}_2(j)$
- $k_j^{i-1} = F_{s_j}(i-1)$
- $k_j^i = F_{s_j}(i)$
- $SK^{i-1} = Dec(k_j^{i-1}, S^{i-1}, hdr^{i-1})$
- $SK^i = Dec(k_j^i, S^i, hdr^i)$

Method 2. This method mitigates the node capture attack as follows. The current node's session-specific private key is derived from the previous session-specific private key using one-way function F_0. Initial key k_j^0 is pre-loaded (and shared with the BC), and the private key for session i is computed by jth node as follows:

$$k_j^i = F_0\left(k_j^{i-1}\right). \tag{2}$$

Although this method is more secure than Method 1, it has some requirements. A crucial aspect is erasing the old session-specific private key from the node's memory. More specifically, after obtaining key k_j^i, previous session-specific key k_j^{i-1} must be permanently erased from its memory. Because keys form *a one-way chain*, the computational overhead increases. In the worst case, the BC and the node must perform i computations to synchronize the key for the ith session.

An adversary that captures node j in session i acquires only session-specific key k_j^i. Let us assume that an adversary has access to the following function:

$$k_j^i = \mathcal{O}_3(i, j), \tag{3}$$

which returns private key k_j^i for the given session number i and node number j. An adversary queries $\mathcal{O}_3(m, j)$ multiple times (where $m \geq i$), and if he can compute k_j^{i-1} effectively, then it leads to a contradiction as computing the previous key k_j^{i-1} is equivalent to inverting pseudorandom function F_0, which is assumed to be impossible. As a result, the adversary can decrypt the current and future sessions but cannot obtain the key(s) for any previous session(s).

However, let us consider an attack in which only the current key of a given node is leaked (without capturing the entire node's state). This threat is realistic, for example, because of some unknown weaknesses in the BE scheme, side-channel attacks, or an implementation bug. After this attack succeeds, the adversary using leaked session-specific key k_j^i based on Equation (2) can compute any arbitrary private key(s) of jth node and the session key(s) of any future session(s), assuming that the node belongs to the privileged set. Consequently, Method 2 is vulnerable to such a threat despite holding the BBP property.

Method 3. This method improves the weakness of the previous method: an attacker capturing just one session-specific private key of a node can compute that node's secret keys of the future. Hence, a secure key update method should exclude such an attack by using a keyed pseudorandom function as follows:

$$k_j^i = F_{s_j}\left(k_j^{i-1}\right). \tag{4}$$

This solution requires the node and the BC to share long-term s_j and initial private key k_j^0. Similar to Method 2, after generating key k_j^i, previous key k_j^{i-1} must be permanently removed from the node's memory. Note that the computational cost is the same as in Method 2.

In Method 3, it is trivial to notice that an adversary cannot compute k_j^i even with an access to k_j^{i-1} without knowing s_j. In other words, to capture the current and the future keys, an attacker must obtain the long-term secret as well. Even with s_j and k_j^i, the attacker cannot compute previous key k_j^{i-1}. Hence, BBP is achieved.

A stronger attacker model is when an adversary observes *output keys* (either independent random values, or values produced by consecutive executions of a key update

function at a node), and the attacker is allowed to dynamically choose when he wants to capture the node [11]. The adversary wins if he can distinguish whether the output keys were produced by the key update function or they were independent random values. Such a notion of security is quite strong and Method 3 cannot defend such an attacker model. For example, an adversary observes two consecutive output keys k_j^{i-1} and k_j^i, and then breaks in. Now with the knowledge of the node's current state (s_j), the adversary can easily check if the keys were produced by the key update function, simply by checking if $F_{s_j}\left(k_j^{i-1}\right)$ and k_j^i are equal.

Method 4. This method is secure even to the attack as presented in the previous text. Key update is performed as the following atomic operation:

$$k_j^i = F_{s_j^{i-1}}(0), \tag{5a}$$

$$s_j^i = F_{s_j^{i-1}}(1). \tag{5b}$$

This method requires two executions of PRF and one initial secret s_j^0 shared between the BC and the node. This shared secret is updated in every session; hence, each private key is generated using a different secret. Consequently, an adversary with consecutive keys cannot determine their origin, because at the moment of the attack he has access to current state $\left(s_j^i, k_j^i\right)$ only. Hence, he is unable to compute the previous states that were used to produce the current keys. Bellare and Yee formally analyzed this method (see Construction 2.2) [11].

Method 4 provides an extremely strong security property. However, the attack as presented above may be impractical and unrealistic for various applications and networks. Furthermore, this solution introduces computational overhead, since every key update operation requires two F_k executions, whereas other solutions only require one.

5. SECURITY ANALYSIS

The major disadvantage of the initial BE scheme as presented in Protocol 1 is its one-time feature. For enhancement, Section 4.3 presents four key update methods with different efficiency and security properties. However, even if the initial BE protocol is secure for one execution, combining the protocol with the key update functions may introduce new attack vectors. For example, we can consider five attacks: (1) one-time exploitation attack in Algorithm 2; (2) single-session-key FS exploitation attack (i.e., an attacker has only one session-specific private key of a privileged node and tries to decrypt the next session); (3) single-session-key BS exploitation attack (i.e., an attacker has only one session-specific private key of a privileged node and tries to decrypt the previous session); (4) BBP attack (i.e., an attacker knows a node's entire state and attempts to decrypt the messages in the previous session); and (5) multi-session-key FS exploitation attack (i.e., an attacker observes multiple keys and distinguishes legitimate keys from random values [11]). The attack resilience of the presented methods in Section 4.3 is summarized in Table I.

Table I. Attack resilience of presented key update methods.

		Method			
		I	**II**	**III**	**IV**
Attack	One-time exploitation attack in Algorithm 2	+	+	+	+
	Single-session-key FS exploitation attack	+	–	+	+
	Single-session key BS exploitation attack	+	+	+	+
	BBP attack	–	+	+	+
	Multi-session-key FS exploitation attack	–	–	–	+

'+' represents resilience to the given attack, and '–' represents that the method is insecure in the face of the given attack.
FS: forward secrecy; BS: backward secrecy; BBP: break-backward protection.

Method 4 (Equation 5) is resilient to all five attacks and hence is the most secure method. However, it consumes twice more computations than any other methods, and the security advantage is achieved for the unrealistic adversarial model (i.e., attack (5)). Method 3 (Equation 4), on the other hand, is resilient to attacks (1)–(4) and hence can be considered as the relaxed method, with a trade-off between security and efficiency. Hence, in the following analysis, we consider Method 3 as the default and recommended solution. By using Method 3 with the initial BE scheme, a session key is protected from various identified attacks. As an effect, even an adversary that captures nodes cannot get the previous session key(s). We now present the formal arguments.

Let us assume that privileged set S consists of nodes with long-term keys $s_1, s_2, \ldots, s_n$ shared with the BC. Suppose that an adversary obtains all keys except one key s_n. Then he or she is still unable to compute anything about s_n before the protocol's execution. The adversary has knowledge of private keys of nodes with id $1, 2, \ldots, n-1$. However, because keys were selected uniformly at random, he or she cannot determine anything about the private key of the node with id n.

Let us further assume that the adversary can capture nodes $1, 2, \ldots, n-1$ and obtain session key k_n^1 (e.g., using a side-channel attack) after the protocol executes once. Now, the adversary's goal is to compute the initial key k_n^0, and we show that this private key after one round of the protocol remains secret with the key update Method 3. Let us assume that such an attack exists and function $\mathcal{O}_4$ given an input $\left(\{s_1, \ldots, s_{n-1}\}, k_n^1, hdr^1\right)$ can compute k_n^0:

$$k_n^0 = \mathcal{O}_4\left(\{s_1, \ldots, s_{n-1}\}, k_n^1, hdr^1\right). \tag{6}$$

Algorithm 4: Security proof for the protocol after a single execution.

$\mathcal{A}(y)$:
$s_1, s_2, \ldots, s_{n-1} = Gen(n-1)$
$k_1^0, k_2^0, \ldots, k_{n-1}^0 = Gen(n-1)$
for $m \in \{1, \ldots, n-1\}$
$k_m^1 = F_{s_m}(k_m^0)$ (Equation 4)
$k_n^1 = y$
$S^1 = \{1, 2, \ldots, n\}$
$SK^1, hdr^1 = Enc(\{k_x | x \in S^1\})$
$x = \mathcal{O}_4(\{s_1, s_2, \ldots, s_{n-1}\}, y, hdr^1)$
return x

Then there exists an adversary (presented in Algorithm 4) that can invert pseudorandom function F_k. Thus, if the method is broken by this adversary, it implies that F_k is inverted, contradicting our assumption that F_k is non-invertible. A similar reasoning can be applied to prove the secrecy of long-term key s_n, as it is ensured by the assumed PRF properties.

The previous argument is for one execution of the protocol and for the secrecy of an initial key. The security of the protocol after many executions (and for the secrecy of next keys) is supported by the following reasoning. Let us assume that there exists function $\mathcal{O}_5$ which, after j protocol executions, can break the method and return key k_n^{j+1}. Given $\left(\{s_1, \ldots, s_{n-1}\}, \{k_1^0, \ldots, k_n^j\}, \{hdr^0, \ldots, hdr^j\}\right)$ as an input, this function computes k_n^{j+1}:

$$k_n^{j+1} = \mathcal{O}_5\left(\{s_1, \ldots, s_{n-1}\}, \{k_n^0, \ldots, k_n^j\}, \{hdr^0, \ldots, hdr^j\}\right) \tag{7}$$

If such $\mathcal{O}_5$ exists, then F_k can be broken by the adversary in Algorithm 5. The adversary without key k and without querying $F_k(x^j)$ can produce the next value of F_k, indicating that he can distinguish F_k from a truly randomly selected function, contradicting the assumption that F_k is a PRF. Thus, an adversary after j rounds cannot compute the next private key of the targeted node, implying that the adversary cannot compute long-term key s_n.

Algorithm 5: Security proof for the protocol executed many times.

$\mathcal{A}(n, j)$:
$s_1, \ldots, s_{n-1} = Gen(n-1)$
$k_1^0, \ldots, k_{n-1}^0, x^0 = Gen(n)$
$S = \{1, 2, \ldots, n\}$
for $l \in \{1, \ldots, j\}$
for $m \in \{1, \ldots, n-1\}$
$k_m^l = F_{s_m}(k_m^{l-1})$ (Equation 4)
$x^l = F_k(x^{l-1})$
$SK^l, hdr^l = Enc(\{k_x | x \in S\}$
$x^{j+1} = \mathcal{O}_5(\{s_1, \ldots, s_{n-1}\}, \{x^0, \ldots, x^j\},$
$\{hdr^0, \ldots, hdr^j\})$
return x^{j+1}

Protocol 1 by itself provides resilience such that an adversary knowing the secrets of all nodes from unprivileged set R cannot compute the session key. Additionally, the private key of each node is chosen randomly. Hence, in order to compute one node's key, collusion of the rest of the users is insufficient. Moreover, this solution achieves FS, BS, and BBP properties with a proper key update method.

Although validating group keying protocols is a research problem itself and hence outside the scope of this paper, we conducted security analysis with a tool for automated validation. Verifying group key protocols is challenging because of their complexity and the size of both states and knowledge sets [12]. Thus, we bounded our reasoning to a fixed number of users and sessions. We also modeled the *Enc* algorithm (Protocol 1) as a pseudorandom function. To perform our analysis, we used On-the-Fly Model Checker (OFMC) and AtSe backends of the AVISPA tool [13]. We chose Automated Validation of Internet Security Protocols and Applications (AVISPA) because it provides the communication model [14] that characterizes WSN, (e.g., adversary can eavesdrop, modify, and block the traffic) and allows us to model node's compromise attack by expressing adversary's knowledge. We confirmed how our methods satisfy the security properties, and the example model for the main property (BBP) is publicly available.†

Another important aspect of the method is the key. Key length is strictly connected with length of prime number q, which is constant and known by only the users and BC. Nodes' private keys are represented as points $(x, y) \in GF(q)^2$, while the group session key is $SK \in GF(q)$. Thus, the effective security level is $\log_2 q$ bits, but not $2\log_2 q$ bits as users' private keys suggest. An adversary launching a brute-force attack can search the session key in $GF(q)$ instead of $GF(q)^2$. Besides the brute-force attack protection, parameter q should be selected in such a way that the probability of collisions is negligible while updating users' private keys.

Our method supports *key's leakage tracing* that allows to identify owners of captured keys in a session. As initial private keys are shared with the BC, it should be possible to find the source of the leakage. However, with the key update method, the complexity increases. An underlying assumption is that the BC does not erase nodes' (at least initial) keys from its memory after the key update process. The computational effort depends on the used key update and the BC's knowledge of a leakage. Table II shows required computations for different methods under two different scenarios. For example, under a brute-force attack, the BC starts from initial keys and tries to find the leaked key by updating the sequence of keys one by one. Such a strategy can be optimized [15,16].

One can consider a variant where the BC may be captured as well. In such a case, a reasonable strategy is to erase old nodes' private keys from the BC's memory and

† http://people.inf.ethz.ch/spawel/broad_enc/model.hlpsl.

Table II. Computational effort needed for leakage tracing when: (a) key is leaked in ith session, or (b) moment of leakage is unknown and the current session is i.

	Key update method			
	I	**II**	**III**	**IV**
(a) Brute-force	$O(n)$	$O(in)$	$O(in)$	$O(in)$
(a) Using [15]	—	$O\left(n\log_2^2 i\right)$	$O\left(n\log_2^2 i\right)$	$O\left(n\log_2^2 i\right)$
(b) Brute-force	$O(in)$	$O(in)$	$O(in)$	$O(in)$
(b) Using [15]	—	$O\left(n\log_2^2 i\right)$	$O\left(n\log_2^2 i\right)$	$O\left(n\log_2^2 i\right)$

$n = |U|$

store the current keys only. Then key update on the BC is performed exactly in the same way as performed in regular nodes. In this setting, even if the BC (or even the entire network) is captured, the adversary cannot decrypt any messages in previous sessions. However, as a consequence of this configuration, leakage tracing becomes infeasible.

6. SCHEME MODIFICATIONS AND IMPLEMENTATION

One major disadvantage of the BE scheme is its lack of effective user management in common scenarios: To remove one user from set S, the BC has to execute the protocol again and send $|S| - 1$ messages. Similar issues arise for adding a new member. Consequently, such a scheme can be considered as effective only when the node's storage overhead becomes a priority, or when the privileged set changes radically every session.

A feasible solution to address this problem is to group nodes into hierarchical structures. Then group keys are first established for small groups, and small-group keys are used for larger-group keys. This process continues until the session key is established for the entire privileged set. Consequently, transforming the proposed BE scheme into a *tree-based* scheme is promising [17]. To establish an initial session key for a privileged set, the BC needs to send $|S| - 1$ messages, and to add or remove a member, the BC needs to send only $O(\log_2 |S|)$ messages, which is significantly better than sending $|S| - 1$ in the original scheme. The storage and computational overheads at nodes are also $O(\log_2 |S|)$. For an adversarial network scenario in Section 2.1, this tree structure can be constructed to optimize operations; for example, nodes that are expected to be captured concurrently form one *sub-tree*. This strategy is shown in Figure 1. The BC first sends headers to establish K_{12}, K_{34}, K_{56}, and K_{78}, then along with these keys headers for K_{1234} and K_{5678} are sent. The last header establishes session key SK. With this structure, even after nodes 5, 6, 7, and 8 are captured, the rest of the network can still use shared key K_{1234} that remains to be unknown to captured nodes 5, 6, 7, and 8.

We implemented the tree-based variant of the scheme because of its efficiency, supporting two security parameters with length 80 and 128 bits. Although these parameters are the characteristics for lightweight cryptography [18], we emphasize that security parameters should be selected carefully, as they depend on multiple factors, such as amount of traffic, characteristics of data, and number of nodes as well as their capabilities. The implementation was tested on both regular node and BC sides. The BC was implemented on a PC machine with Intel i5 Core 3.2 GHz and 8GB of RAM. For a regular node, we used Alix.3d [19] equipped with 500 MHz AMD Geode LX800 and 256MB of RAM. We utilized GMPlib [20] for mathematical operations and OpenSSL [21] for cryptographic operations (Advanced Encryption Standard (AES) cipher [22] was selected as pseudorandom function F_k). In terms of security parameters, we selected to implement four groups with different numbers of privileged users, and each group was composed of 8, 32, 256, and 1024 nodes.

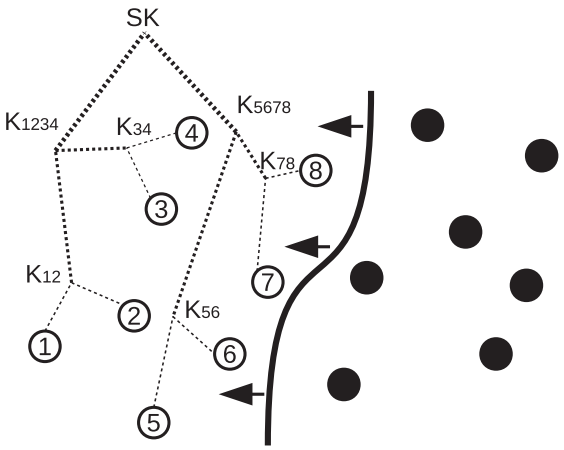

Figure 1. Tree-based hierarchy where some nodes have greater probability of being captured.

Performance results are presented in Table III using required CPU time (in milliseconds) for a given operation on both regular node and the BC. The first operation, denoted as *Init*, is for the initial session key establishment for $|S|$ users. This operation from the privileged set forms a balanced binary tree structure. The *Add* operation is for adding a new node to the tree structure, while *Del* denotes removing a node from the privileged set. Add_{node}^{max} and Del_{node}^{max} results are for the worst case computations, which must be performed by a newcomer or the sibling of a removed node. However, because of tree-structure properties, most of the nodes from S perform fewer computations. Minimum and average computation times are denoted as $Add_{node}^{min}, Del_{node}^{min}$, and $Add_{node}^{avg}, Del_{node}^{avg}$, respectively.

Our implementation results show that the construction is efficient in the common case. The initial group establishment (*Init*) is the most expensive operation. For 1024 nodes with 128-bit-long keys, forming a tree and establishing a session key require every node to perform computations for approximately 221 ms, while this operation on the BC takes only 11 ms. After this initialization, the management of nodes' membership is effective. To add a node to

Table III. Performance results (expressed in milliseconds) for presented protocol in tree-based setting.

$\lceil \log_2 q \rceil$	80				128			
$\|S\|$	8	32	256	1024	8	32	256	1024
$Init_{node}$	1.11	4.96	40.82	163.76	1.49	6.66	55.52	221.40
$Init_{BC}$	0.05	0.23	1.99	8.00	0.07	0.33	2.71	11.17
Add_{node}^{max}	0.65	1.00	1.44	1.75	0.88	1.32	1.91	2.36
Add_{node}^{min}	0.33	0.33	0.33	0.33	0.44	0.44	0.44	0.44
Add_{node}^{avg}	0.46	0.49	0.50	0.50	0.64	0.68	0.69	0.69
Add_{BC}	0.03	0.04	0.07	0.08	0.04	0.06	0.09	0.16
Del_{node}^{max}	0.40	0.56	1.11	1.44	0.44	0.88	1.48	1.92
Del_{node}^{min}	0.25	0.25	0.25	0.25	0.32	0.32	0.32	0.32
Del_{node}^{avg}	0.36	0.48	0.50	0.50	0.39	0.68	0.69	0.69
Del_{BC}	0.01	0.03	0.05	0.07	0.02	0.04	0.07	0.09

Table IV. Computation and communication cost of the protocols.

	Operations	Msgs sent	Msgs received
GDH_{init}	$\|S\|$	1	$\|S\| - 1$
GDH_{add}	$\|S\| + 3$	1	$\|S\|$
GDH_{del}	$\|S\| - 1$	1	0
$TGDH_{init}$	$2 \log_2 \|S\|$	$\|S\|$	$\|S\|$
$TGDH_{add,del}$	$2 \log_2 \|S\| - 2$	0	1
STR_{init}	$\|S\| - 1$	$\|S\| - 1$	0
STR_{add}	4	2	1
STR_{del}	$\frac{3\|S\|}{2} + 2$	0	1
Our $scheme_{init}$	$\|S\| - 1$	0	$\|S\| - 1$
Our $scheme_{add,del}$	$\log_2 \|S\|$	0	$\log_2 \|S\|$

Column *operations* describes number of operations required to accomplish given action.

a set consisting of 1024 privileged nodes, the newcomer must perform computations for 2.4 ms. Revocation of a node from the same set can take 1.9 ms at maximum. The same operations on the BC take 0.115 and 0.093 ms, respectively. Lowering security parameter q also reduces the computation effort. However, increasing the security parameter by 60% (from 80 to 128 bits) increases the computational effort by 35% only. Thus, there is an incentive to employ a better security parameter.

7. COMPARATIVE EFFICIENCY ANALYSIS

We compared the efficiency of our scheme with other prominent protocols: Group Diffie-Hellman (GDH) [23] (version 3, which is the most efficient one), Tree-based GDH (TGDH) [24] (improvement of GDH), and STR [25]. The performance results are shown in Table IV, where our scheme is presented in tree-based setting. Results are given for the worst case (i.e., number of messages and operations required by a node with the highest overhead [24]) and for three main actions: (1) initialization (one-time operation per group of users), (2) addition (when a node joins the privileged set), and (3) deletion (when a node leaves the privileged set). Note that the number of operations required to accomplish actions depends on the scheme. For example, the number of modular exponentiation is considered for Diffie–Hellman-based protocols, while the number of pairwise key establishment is considered for our scheme. For hierarchical approaches, we assume that the privileged set forms a balanced tree.

While theoretical comparison may not show the actual performance, we built the simulation environment dedicated for WSN networks. We prototyped and simulated all compared schemes using RELIC [26] library and AVR simulation and analysis framework Avrora [27]. These two tools allowed us to (a) employ AVR-optimized

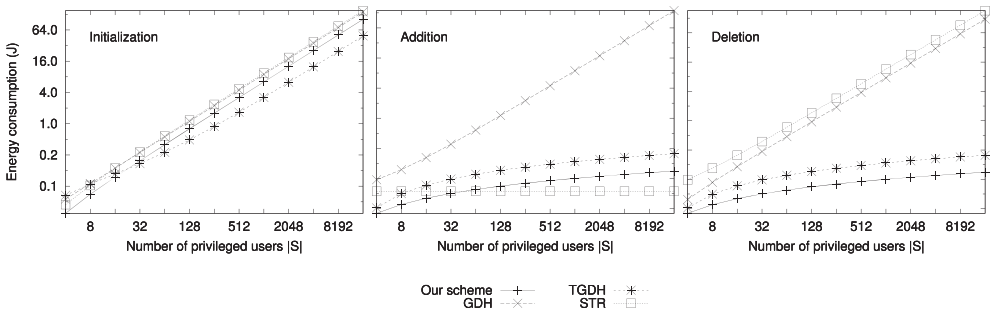

Figure 2. Comparative simulation results for three types of operations (note that both axes are logarithmic scales).

cryptography for prototyping compared schemes and (b) simulate and analyze energy consumption on such a platform (through Avrora's *energy monitor*). In tests, we replaced modular exponentiations with significantly faster operations over elliptic curve (curve *NIST K163* provided by RELIC). We used a 158-bit-long prime (*BN 158* provided by RELIC) for our scheme. In this configuration, multiplication over elliptic curve, pairwise key establishment, and key derivation of Method 3 consumes 7.45, 3.88, and 0.22 mJ, respectively. As previously, we analyzed the worst case scenario, and the performance results are presented in Figure 2.

Both theoretical and simulation comparisons confirm efficiency of our proposal, even for large-scale deployment. Only TGDH achieves similar results when adding or removing nodes to/from large privileged set. STR has the best efficiency for addition; however, it does not scale when a nodes needs to be removed from large set. It is also important to note that no other scheme provides BBP property or its equivalent besides our scheme.

8. RELATED WORK

For secure group key establishment, most of the proposed solutions address a standard adversary model or their constructions deploy cryptography, which is inefficient in networks with limited capabilities. For example, He *et al.* [28] present a nice survey of key management schemes for sensor networks, but many of the described schemes satisfy only standard security requirements (i.e., without assuming a strong adversary model). Some researchers also propose prominent approaches that provide most of the introduced properties, except BBP property [24,25]. Several researchers propose key management schemes assuming the node capture attack [29,30]. For example, Guo and Qian [29] present the pairwise key update protocol that is motivated by the results of Chadha *et al.* [31]. Also, Zhang *et al.* propose a scheme that aims to prevent the node-capturing attacker for hierarchical sensor networks using perturbation polynomical properties for the key update mechanism [32]. Divya and Thirumurugan introduce a dynamic key management scheme that mitigates node capture and collusion attacks based on Hamming distance [30].

Schwenk describes attacks on one-time broadcast encryption schemes and the defense against such attacks [10]. Although one of his methods to secure broadcast encryption is similar to Method 1 in Section 4.3, the main distinction is that he utilizes random numbers instead of consecutive session numbers. Schwenk also provides formal arguments for the final construction, but his analysis encompasses the standard attacker model and few security properties only. Szalachowski and Kotulski improve Schwenk's results by introducing a secure scheme in the face of node capture attacks [7]. Their solution utilizes a hash-chain-based key update scheme and addresses security of the ciphertext transmission (from the BC to nodes), providing confidentiality, authentication, and weak freshness, and the security of such transmission is further studied [33].

Bellare and Yee study forward security in private key cryptography [11]. They introduce a new security model and present cryptographic constructions that are secure in their model (e.g., PRF, message authentication codes, and encryption). In the context of our work, their forward secure pseudorandom bit generator is especially interesting, because it provides the similar properties of Method 4 in Section 4.3.

Mauw *et al.* [34] provide a set of cryptographic measures to provide forward security in WSN environments. Their paper focuses on *node-to-BC* communication and their methods address confidentiality, authentication, and weak freshness. For ensuring the security of private keys in the face of node capture attacks, they also introduce a key update function.

Detecting node capture attacks is outside the scope of this paper, but such a detection mechanism would be a great supplement. For example, Conti *et al.* [35] propose two detection solutions in mobile WSN, given an assumption that an adversary removes the captured node from the network and then tamper with it.

9. CONCLUSIONS

This paper presents an efficient framework for secure broadcast in the presence of a strong adversary that can physically capture nodes. Our construction consists of two main elements. First, we modify the broadcast encryption protocol that is required for a secure group key establishment. Second, we present several key management approaches that improve the initial broadcast encryption protocol and provide certain security properties even if an attacker captures a node. These two elements construct the secure and efficient final solution.

Our solutions provide the high level of security guarantees for hardware-limited and distributed networks. For example, an eavesdropper cannot obtain the session key or the private key of any node. Furthermore, even if an adversary has additional knowledge of all the keys from the revoked node set, his advantage stays the same. In other words, the adversary cannot obtain previous, current, and any of the future session keys. By capturing a node, the strongest action an adversary can launch is obtaining the state of the (even privileged) node, and learning the current session key (as well as the future keys if the attack remains undetected and the captured node is privileged). However, the attacker cannot obtain the keys in the previous sessions, which is a significant advantage. Our solutions support other features such as the key's leakage tracing, which can be optimized for the key update schemes.

All our approaches attempt to move the computational and storage burden to the BC, which is a reasonable strategy when regular nodes within the network are hardware-limited. We introduce two topological variants of the protocol: (1) The *flat* scheme is efficient when the

DOI: 10.1002/sec

privileged set changes significantly and/or when the storage overhead on regular nodes is low; and (2) The *tree-based* variant increases the storage overhead but decreases the computational and transmission overheads given slightly different privileged sets over two consecutive sessions. Based on our implementation of the tree-based variant, our evaluation and performance results on both BC and regular nodes prove that our solution provides efficiency along with security.

ACKNOWLEDGEMENT

This work is supported by the National Science Center (NCN), under grant number DEC-2011/01/N/ST7/02995.

REFERENCES

1. Kavitha T, Sridharan D. Security vulnerabilities in wireless sensor networks: a survey. *Journal of Information Assurance and Security* 2010; **5**(1): 31–44.
2. Perrig A, Stankovic J, Wagner D. Security in wireless sensor networks. *Communications of the ACM* 2004; **47**: 53–57.
3. Tague P, Poovendran R. In Proceedings of Modeling node capture attacks in wireless sensor networks. In *Proceedings of 2008 46th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, 2008; 1221–1224.
4. Wander AS, Gura N, Eberle H, Gupta V, Shantz SC. Energy analysis of public-key cryptography for wireless sensor networks. In *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications*, PerCom '05, IEEE, Kauai, HI, 2005; 324–328.
5. Phan DH, Pointcheval D, Strefler M. Security notions for broadcast encryption. In *Proceedings of the 9th International Conference on Applied Cryptography and Network Security*, ACNS'11. Springer-Verlag: Berlin, Heidelberg, 2011; 377–394.
6. Perrig A, Canetti R, Tygar JD, Song D. The TESLA broadcast authentication protocol. *RSA CryptoBytes* 2002; **5:2**(Summer/Fall): 2–13.
7. Szalachowski P, Kotulski Z. One-time broadcast encryption schemes in distributed sensor networks. *International Journal of Distributed Sensor Networks* 2012, Article ID 536718, 9 pages, DOI:http://dx.doi.org/10.1155/2012/536718.
8. Berkovits S. How to broadcast a secret. In *Proceedings of the10th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'91, Brighton, UK, 1991; 535–541.
9. Shamir A. How to share a secret. *Communications of the ACM* 1979; **22**: 612–613.
10. Schwenk J. How to securely broadcast a secret. In *Secure Information Networks: Communications an Multimedia Security*, Vol. 2, Preneel B (ed). Springer-Verlag US, 1999; 247–257.
11. Bellare M, Yee B. Forward-security in private-key cryptography. In *Proceedings of the 2003 RSA Conference on the Cryptographers' Track,* CT-RSA '03. Springer-Verlag: Berlin, Heidelberg, 2003; 1–18.
12. Martina JE, Paulson LC. Verifying multicast-based security protocols using the inductive method. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, ACM, Coimbra, Portugal, 2013; 1824–1829.
13. Armando A, Basin D, Boichut Y, *et al.* The AVISPA tool for the automated validation of internet security protocols and applications. *In Proceedings of the 17th International Conference on Computer Aided Verification.* Springer-Verlag Berlin Heidelberg, Scotland, UK, 2005; 281–285.
14. Dolev D, Yao A. On the security of public key protocols. *IEEE Transactions on Information Theory* 1983; **29**(2): 198–208.
15. Coppersmith D, Jakobsson M. Almost optimal hash sequence traversal. In *Proceedings of the 6th International Conference on Financial Cryptography,* FC '02. Springer-Verlag: Berlin, Heidelberg, 2003; 102–119.
16. Cairns K, Gamage T, Hauser C. Efficient targeted key subset retrieval in fractal hash sequences. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security,* CCS '13. ACM: New York, NY, USA, 2013; 1273–1284.
17. Sherman AT, McGrew DA. Key establishment in large dynamic groups using one-way function trees. *IEEE Transactions on Software Engineering. IEEE Press* 2003; **29**(5): 444–458.
18. ISO/IEC. *ISO/IEC 29192. Information Technology Security Techniques Lightweight Cryptography*. ISO/IEC, 2012.
19. PC Engines. Available from: http://www.pcengines.ch/alix3d2.htm [Accessed 4 June 2015.]
20. GMPlib. Available from: http://gmplib.org/ [Accessed 4 June 2015].
21. OpenSSL. Available from: http://www.openssl.org/ [Accessed 4 June 2015].
22. Daemen J, Rijmen V. *The Design of Rijndael*. Springer-Verlag New York, Inc.: Secaucus, NJ, USA, 2002.
23. Steiner M, Tsudik G, Waidner M. Diffie–Hellman key distribution extended to group communication. In *Proceedings of the 3rd ACM Conference on Computer and Communications Security*, ACM, New Dehli, India, 1996; 31–37.

24. Kim Y, Perrig A, Tsudik G. Tree-based group key agreement. *ACM Transactions on Information and System Security (TISSEC)* 2004; **7**(1): 60–96.
25. Kim Y, Perrig A, Tsudik G. Communication-efficient group key agreement. In *Proceedings of the IFIP TC11 16th Annual Working Conference on Information Security: Trusted Information: The New Decade Challenge*. Kluwer, B.V. Paris, France, 2001; 229–244.
26. Aranha DF, Gouvêa CPL. RELIC is an Efficient LIbrary for Cryptography. Available from: https://github.com/relic-toolkit/relic [Accessed 4 June 2015].
27. Titzer BL, Lee DK, Palsberg J. Avrora: scalable sensor network simulation with precise timing. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks.* IEEE Press, Los Angeles, CA, 2005; 67.
28. He X, Niedermeier M, de Meer H. Dynamic key management in wireless sensor networks: a survey. *Journal of Network and Computer Applications* 2013; **36**(2): 611–622.
29. Guo S, Shen AN. A compromise-resilient pair-wise rekeying protocol in hierarchical wireless sensor networks. *Computer Systems Science and Engineering* 2010; **25**(6): 315–326.
30. Divya R, Thirumurugan T. A novel dynamic key management scheme based on hamming distance for wireless sensor networks. In *Proceedings of the International Conference on Computer, Communication and Electrical Technology (ICCCET).* IEEE, Tamil Nadu, India, 2011; 181–185.
31. Chadha A, Liu Y, Das SK. Group key distribution via local collaboration in wireless sensor networks. In *Proceedings of the Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks,* IEEE, Santa Clara, CA, 2005; 46–54.
32. Zhang W, Tran M, Zhu S, Cao G. A random perturbation-based scheme for pairwise key establishment in sensor networks, In *Proceedings of the 8th ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2007*, ACM, Montreal, QC, Canada, 2007; 90–99.
33. Szalachowski P, Perrig A. Lightweight protection of group content distribution. In *Proceedings of the 1st ACM Workshop on IoT Privacy, Trust, and Security*, IoTPTS '15. Singapore, Republic of Singapore, ACM: New York, NY, USA, 2015, 35.
34. Mauw S, Vessem IV, Bos B. Forward secure communication in wireless sensor networks. In *SPC 06: Proceedings of the 3rd International Conference on Security in Pervasive Computing*, Lecture Notes in Computer Science, York, United Kingdom; 32–42.
35. Conti M, Di Pietro R, Mancini LV, Mei A. Emergent properties: detection of the node-capture attack in mobile wireless sensor networks. In *Proceedings of the First ACM Conference on Wireless Network Security,* WiSec '08. ACM: New York, NY, USA, 2008; 214–219.