Definition and Development of a *Topology-Based* Cryptographic Scheme for Wireless Sensor Networks

Stefano Marchesani, Luigi Pomante,
Marco Pugliese, and Fortunato Santucci

Center of Excellence DEWS, Università degli Studi dell'Aquila, L'Aquila, Italy
{stefano.marchesani,luigi.pomante,fortunato.santucci}@univaq.it,
marco.pugliese@ieee.org

Abstract. A Wireless Sensor Network (WSN) is a versatile and distributed sensing system that is able to support a wide variety of application domains. One of the most important issue in WSN design is to guarantee the reliability of the collected data which involves in turn security issues across wireless links. This paper deals with the cryptographic aspects in the broader field of security in WSNs. In particular, moving from some previous advances in our research activity, this manuscript proposes a novel cryptographic scheme that is compliant to security requirements that may arise from real-world WSN applications and reports some details about an implementation in *TinyOS* that we have carried for experimental validation. The proposed scheme, called *TAKS2*, exploits benefits from *Hybrid Cryptography* to handle resource constraints and allows to generate *topology authenticated keys* to provide increased robustness to the scheme itself.

Keywords: WSN security, cryptographic scheme, hybrid cryptography, topology authenticated key.

1 Introduction and Contribution

In this paper, we propose a novel scheme to generate *topology authenticated keys* for handling cryptographic aspects in resource constrained deployments of Wireless Sensor Networks. We then describe the implementation of the proposed scheme in *TinyOS*, an operating system for a variety of families of sensor nodes [18], and its real deployment and testing on some of *MICAz* sensor nodes.

The proposed scheme, called *TAKS2*, exploits benefits from both symmetric and asymmetric schemes (*Hybrid Cryptography*) but here only *partial components* of symmetric keys are pre-distributed and not the keys as all. The cryptographic scheme presented in this paper is an upgrade of *TAKS*, that was earlier presented in [15] and later refined in [14] wherein we extended the scheme to *Elliptic Curve Cryptography* (ECC). Upgrades we propose in this paper are related to simplifications in the key management protocol and reductions in memory usage.

M. Zuniga and G. Dini (Eds): S-Cube 2013, LNICST 122, pp. 47–64, 2013.

The remainder of this paper is organized as follows. In Section 2, an overview of state-of-art about cryptography applied to WSN is provided along with current pending issues, and a brief reference to the main concepts of TAKS are reported. Section 3 describes the cryptographic scheme and its main features are compared with the corresponding ones of TAKS. In Section 4, the scheme is formally defined and its components are rigorously described. In Section 5, the security of the proposed scheme is formally analyzed. In Section 6, implementation issues over sensor nodes, cost analysis and scheme testbed are discussed. In Section 7, some conclusive comments and future works are reported as well.

2 Background and Motivations

Providing security in traditional networks often means using asymmetric encryption. In general contexts, the ever increasing amount of available resources in terms of computation, memory and power supply makes it possible to ignore the main disadvantage of this strategy: indeed, the robustness of asymmetric algorithms is highly dependent on the size of the keys that in turn affects the complexity of the algorithms. Most recent studies have led to the definition of ECC [6]. Among those alternatives that resort on public keys, this solution appears as the best choice in terms of execution times and key size in order to meet some given requirements [19].

However, whenever computation resources become tightly constrained (e.g. in WSNs), the role of symmetric cryptography becomes again relevant. The most important disadvantage is related to *key management*, since encryption and decryption algorithms are lighter than those based on asymmetric key. Nevertheless, this problem is one of the most addressed ones in the literature [4]. There are *pair-wise key pre-distribution* solutions, that are based on deterministic pre-distribution of keys for each pair of nodes. The trivial solution consists in distributing a key for each pair of nodes, eventually with the same key for the entire network. Other *random pair-wise key schemes* are based on storing only a subset of all possible keys in each node [5]. To communicate with each other, every node needs to negotiate a key with its peer, randomly selecting one key in its subset. If location of nodes is known, it is possible to simplify the last algorithm by providing to each node only the keys for the actual neighbors [9]. Other techniques are based on *cluster pre-distribution*: therefore, in each cluster different keys are used [8] and keys can be build specifically for each pair of nodes according to the nodes composing the cluster itself. Instead, the *master key pre-distribution* requires that a master key is distributed in the entire network and that nodes use a combination of it and previous exchanged *nonces*. Finally, the idea behind the *key matrix based dynamic key generation* [3] is to distribute rows and columns of matrices, whose product is a symmetric matrix, as public and private key, respectively; by multiplying rows and columns, a pair of nodes may produce the same key to encrypt their messages.

With respect to the previous solutions, the scheme proposed in this paper does not rely on pre-distribution of keys in nodes, but it is rather based on

their dynamic generation moving from partial components stored in nodes as in [15]. Through computationally inexpensive operations, a node can compute the decrypt/encrypt key in a single phase without any need of setup/negotiation. Moreover, as will be shown in next section, the stored partial components are defined such that secure communications among that nodes are allowed only if their topology is compliant to the *planned network topology*. This motivates the names TAK (*Topology Authenticated Key*) and TAKS (*Topology Authenticated Key generation Scheme*). We define qualitatively planned network topology as the network topology that is planned by a service manager (the *planner*) to satisfy some service requirements. We also define *eligible neighbor nodes* of a node those nodes with which the node may be authorized to communicate. According to the previous definition, the planned network automatically gets the attribute of *certified network topology*, where the certification authority is the planner itself.

3 Description of the Cryptographic Scheme

To describe the cryptographic scheme following definitions are needed:

- *public*: any information anyone can access (attackers included);
- *restricted*: any information any node in the network can access;
- *private*: any information only a single node in the network can access;
- *secret*: any information only the planner can access.

The proposed scheme requires an offline definition of some parameters (i.e. partial components). We call this set of parameters *Local Configuration Data* (LCD) which define the physical or logical topology configured in each node: topology can be physical or logical depending on the scheme is applied on physical/MAC layer or upper layers respectively. LCD includes:

- *Local Key Component* (*Loc.Key.Comp.*);
- *Transmit Key Component* (*Trans.Key.Comp.*);
- *Local Planned Topology* (*Loc.Pld.Top.*) i.e. a set of *Topology Vectors* in one-to-one relationship with eligible neighbor nodes.

The security of proposed scheme is based on the confidentiality of the information used to generate the keys: Local Key Component and Transmitted Key Component are both private and they are calculated from deployment parameters that are secret. Following we introduce the scheme.
We put q large prime such that $q \gg N$ where N is the total number of nodes in the network. The corresponding keys length will be approximately $\log_2 q$. Let U be a vector space over $GF(q)$ where the generic vector $\overline{u} \in U$ is represented with a 3-pla (u_x, u_y, u_z) of vector components elements in $GF(q)$.
Let $TAK()$ be a function satisfying the following requirements:

R1. it must be a surjective function and $TAK(\overline{u}, \overline{u}') \neq 0, \forall \overline{u}, \overline{u}' \in U$;

R2. $TAK(\overline{u}, h(\overline{u}')) = TAK(\overline{u}', -h(\overline{u})), \forall \overline{u}, \overline{u}' \in U$, where $h()$ is an arbitrary vector function in U;
R3. $TAK(\alpha\overline{u}, \overline{u}') = TAK(\overline{u}, \alpha\overline{u}') = \alpha TAK(\overline{u}, \overline{u}'), \forall \overline{u}, \overline{u}' \in U$ and $\alpha \in GF(q)$;

$TAK(\cdot)$ is the function used to generate TAKs.
Let $g(p, v)$ be a function satisfying the following requirements:

R4. it must be a surjective function;
R5. $g(p, v) = 0$ only for a predefined set of distinct values of p, v.

$g(\cdot)$ is the function used to *verify message authenticity.*
According to the *Kerchoff principle* the explicit expressions for both $TAK(\cdot)$ and $g(\cdot)$ are public.
Fig.1 reports the conceptual representation of the proposed scheme. Let $\sigma(i)$ be the set of eligible neighbors of node n_i, we define the Local Planned Topology of node n_i as $T(i) = \{Trans.Key.Comp._{\sigma(i)}\}$. It is worth noting that each node also stores its own Transmit Key Component that we denote as $Trans.Key.Comp._i$ for node n_i.
If n_i wants to communicate with n_j, it has to generate a random value $\alpha \in GF(q)$ and to build a message as concatenation of:

- the *cipher text* (c) produced by a symmetric encryption algorithm $Encr(\cdot)$ with $\alpha TAK(Loc.Key.Comp._i, Trans.Key.Comp._j)$ as key;
- the *deciphering information* (d) where $d \in U$ and $d = -\alpha Trans.Key.Comp._i$;
- the *message authentication code* (τ) associated to the cipher text using any *cryptographic hash function* (denoted as $MAC(\cdot)$) with key equals to $\alpha TAK(Loc.Key.Comp._i, Trans.Key.Comp._j)$.

When n_j receives the message, it has to calculate a pair-wise key to decrypt it. It computes its own key as $TAK(Loc.Key.Comp._j, d)$. If $Loc.Key.Comp._i$, $Loc.Key.Comp._j \in U$ and $Trans.Key.Comp._i = h(Loc.Key.Comp._i)$ and $Trans.Key.Comp._j = h(Loc.Key.Comp._j)$, we have that:

$$\begin{gathered} \alpha TAK(Loc.Key.Comp._i, Trans.Key.Comp._j) = \\ \alpha TAK(Loc.Key.Comp._j, -Trans.Key.Comp._i) = \\ TAK(Loc.Key.Comp._j, -\alpha Trans.Key.Comp._i) = \\ TAK(Loc.Key.Comp._j, d) \end{gathered}$$

So n_j can correctly decrypt the message and $TAK(Loc.Key.Comp._j, d)$ is validated as TAK. A crucial point is how n_j can recognize its calculated key as a symmetric TAK: this is done by verify message authenticity function $g(\cdot)$ that has to return zero only if encryption and decryption keys are identical. This is established using τ information.
The reference TAKS description can be found in [15]. Here we will deal with the upgrades to TAKS, we denoted as TAKS2, that follow:

- In TAKS2, topology vectors in a node now coincide with the Transmit Key Components of its eligible neighbours. Therefore less memory is needed in each node to store static information.

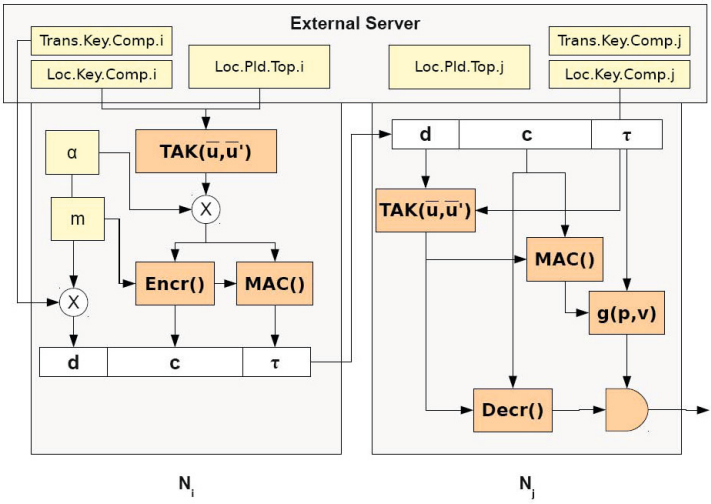

Fig. 1. TAKS2 scheme description

- In TAKS2, the transmission protocol is 1-phase (i.e. there is no need of other party response message to proceed) as there is no need of a prior exchange of the Transmit Key Components between nodes (as occurs in TAKS): each transmission contains the ciphered text (c), the authentication tag and the ephemeral Transmit Key Component of the transmitter. Any node which receives this message can check to be the right recipient and message integrity. The ephemeral Transmit Key Component is defined as the Transmit Key Component multiplied by a one-shot random value.
- In TAKS2, the *Secret Share* (SS) for each eligible node pair is given by TAK multiplied by a one-shot random value (in TAKS is coincident with TAK): therefore in TAKS2 also SS is a one-shot random value for each eligible node pair and security level gets enhanced (in TAKS only a SS value for each eligible node pair, hence each node must store several SS according to the number of eligible neighbours).
- In TAKS2, authentication is performed by a standard authentication function.

The main drawback in TAKS2 is the ephemeral Transmitted Key Component to be transmitted each time. This increases energy consumption per transmission and it can turn to a problem in case of large data transmission rates and large key size. In monitoring applications, transmission rates are related to sampling rates on sensor boards, which depend on the dynamics of the monitored system: if large transmission rates are needed, key size (hence vector size) should be reduced without degrading security: in this occurrence ECC facilities should be included into TAKS2 [14].

4 Formal Apparatus

Building blocks of the proposed scheme are:

1. Hybrid key cryptography
2. Network topology authentication.

4.1 Hybrid Key Cryptography

Let nodes n_i and n_j be a pair. The following definitions are assumed:

a) Let $A, M, KL, KT \in U$ be vector fields.
b) Elements in A are such that for the generic couple $\overline{a}_i, \overline{a}_j \in A$ then $\overline{a}_i \times \overline{a}_j \neq 0$ and fixed a vector $\overline{m} \in M$, then $\overline{m} \cdot (\overline{a}_i \times \overline{a}_j) \neq 0$. This information is secret.
c) Let $b \in GF(q)$ be a scalar not generator in $GF(q)$. This information is secret.
d) Let $f(.) = kb^{\overline{m} \cdot (.)}$ be a scalar function where $\overline{m} \in M$ satisfied (b) and $k \in GF(q)$ is an arbitrary constant.
e) Let $\overline{c} \in U$ be a vector. This information is secret.
f) Let be $\overline{s}_i = \overline{m} f(\overline{a}_i)$ and $\overline{s}_j = \overline{m} f(\overline{a}_j)$ with $k = 1$. Let $\overline{k}_{li}, \overline{k}_{lj} \in KL \subseteq U$ and $\overline{k}_{ti}, \overline{k}_{tj} \in KT \subseteq U$ be defined as:

$$\begin{cases} \overline{k}_{li} = \overline{a}_i k b^{\overline{m} \cdot \overline{a}_i} \\ \overline{k}_{ti} = \overline{s}_i \times \overline{a}_i \end{cases} \qquad \begin{cases} \overline{k}_{lj} = \overline{a}_j k b^{\overline{m} \cdot \overline{a}_j} \\ \overline{k}_{tj} = \overline{s}_j \times \overline{a}_j \end{cases}$$

g) Let $\overline{k}_l \in KL \subseteq U$ and $\overline{k}_t \in KT \subseteq U$ respectively define the Local Key Component and Transmit Key Component in a node. Both components are private.
h) Expressions for $\overline{k}_l$, $\overline{k}_t$ and $f(.)$ are public.

Pair-Wise TAKS. This section deals with the generation of pair-wise TAK or, in other words, a TAK shared in a pair of sensor units.

Theorem 1 (Pair-wise TAKS2 Generation). *Let n_i and n_j be a node pair. Fix $\overline{m}$,$\overline{c}$,b and be $\overline{a}_i, \overline{a}_j \in A$ a generic couple of elements in A compliant to definitions (b) and be $f(.)$ defined as (d) and α a random value in $GF(q)$. If expressions for $\overline{k}_{li}$,$\overline{k}_{ti}$ and for $\overline{k}_{lj}$,$\overline{k}_{tj}$ are the same as (f) then*

$$TAK = \alpha TAK(\overline{k}_{li}, \overline{k}_{tj}) = \alpha TAK(\overline{k}_{lj}, -\overline{k}_{ti}) = \alpha \overline{k}_{li} \cdot \overline{k}_{tj} = -\alpha \overline{k}_{lj} \cdot \overline{k}_{ti}$$

Proof. The proof is straightforward. Applying the definition of TAK_i:

$$TAK_i = \alpha TAK(\overline{k}_{li}, \overline{k}_{tj}) = \alpha \overline{k}_{li} \cdot \overline{k}_{tj} = \alpha \overline{a}_i f(\overline{a}_i) \cdot (\overline{s}_j \times \overline{a}_j) = \alpha \overline{a}_i k b^{\overline{m} \cdot \overline{a}_i} \cdot (\overline{m} b^{\overline{m} \cdot \overline{a}_j} \times \overline{a}_j) = \alpha k b^{\overline{m} \cdot (\overline{a}_i + \overline{a}_j)} \overline{a}_i \cdot (\overline{m} \times \overline{a}_j) = \beta \overline{a}_i \cdot (\overline{m} \times \overline{a}_j)$$

Applying the definition of TAK_j:

$$TAK_j = \alpha TAK(\overline{k}_{lj}, -\overline{k}_{ti}) = -\alpha \overline{k}_{lj} \cdot \overline{k}_{ti} = -\alpha \overline{a}_j f(\overline{a}_j) \cdot (\overline{s}_i \times \overline{a}_i) = -\alpha \overline{a}_j k b^{\overline{m} \cdot \overline{a}_j} \cdot (\overline{m} b^{\overline{m} \cdot \overline{a}_i} \times \overline{a}_i) = -\alpha k b^{\overline{m} \cdot (\overline{a}_j + \overline{a}_i)} \overline{a}_j \cdot (\overline{m} \times \overline{a}_i) = -\beta \overline{a}_j \cdot (\overline{m} \times \overline{a}_i)$$

Exploiting the vector algebra property $\overline{a} \cdot (\overline{s}' \times \overline{a}') = \overline{s}' \cdot (\overline{a}' \times \overline{a})$, we have:

$$TAK_i = \beta \overline{a}_i \cdot (\overline{m} \times \overline{a}_j) = \beta \overline{a}_j \cdot (\overline{a}_i \times \overline{m}) = -\beta \overline{a}_j \cdot (\overline{m} \times \overline{a}_i) = TAK_j \qquad \square$$

In TAKS2, the transmitter TAK is defined as the scalar product between the Local Key Components and the Topology Vector associated to the destination node, while the receiver TAK is defined as the scalar product between the Local Key Component and the Transmit Key Component (in [15] TAK is defined as the squared scalar product) so that the key space gets enhanced ($\sim 50\%$). Furthermore, fixed $\overline{m}$,$\overline{c}$,b and be $\overline{a}_i, \overline{a}_j \in A$ the following properties hold:

1. Always $TAK \neq 0$ being $\overline{m} \cdot (\overline{a}_i \times \overline{a}_j) \neq 0$ from (b) and $f(.) \neq 0$ from R1.
2. Elements in KL are distinct being $\overline{k}_{li} \parallel \overline{a}_i$ and $\overline{k}_{lj} \parallel \overline{a}_j$ with $\overline{a}_i \times \overline{a}_j \neq 0$. Hence $\overline{k}_{li} \times \overline{k}_{lj} \neq 0$.
3. Elements in KT are distinct being $\overline{k}_{ti} \parallel \overline{m} \times \overline{k}_{li}$ and $\overline{k}_{tj} \parallel \overline{m} \times \overline{k}_{lj}$ and $\overline{k}_{li} \times \overline{k}_{lj} \neq 0$.
4. Key components in a node are distinct being for generic node i $\overline{k}_{li} \cdot \overline{k}_{tj} = 0$ because $\overline{k}_{li} \parallel \overline{a}_i$ and $\overline{k}_{ti} \parallel \overline{m} \times \overline{a}_i$.

Cluster-Wise TAKS. This section deals with the generation of *cluster* TAK or, in other words, a common TAK shared in a group of sensor units or cluster. Node clustering is commonly considered as one of the most promising techniques for dealing with the maximization of WSN lifetime. In a clustered WSN, the sensor units are grouped into a set of disjoint clusters: each cluster has a designated leader, the so-called *cluster head* (CH). Nodes in one cluster do not transmit their gathered data directly to the sink, but only to their respective cluster head. Accordingly, the cluster head is responsible for:

- coordination among the cluster nodes and aggregation (i.e. compression) of their data, and
- transmission of the aggregated data to the sink, directly or via multi-hop transmission (for more, see [11]).

In [11] is provided an explicit analysis of node clustering in WSNs and it is proved that the condition that ensures superior performance of clustered WSNs is that the formed clusters lie within the isoclusters of the monitored phenomenon. An isocluster is an area consisting of points that have the same value or lie within a certain limited value range: isocluster is a key concept also for data and alarm aggregation in anomaly detection logic in monitoring applications running over WSN. There are lots of clustering algorithms available from literature (e.g. [1]) each one according to specific aggregation metrics: in [16] application-oriented metrics have been considered.

The vector algebra approach used in TAKS and TAKS2 gives us the chance to generate both pair-wise and cluster-wise cryptographic keys with the same scheme: this is due the definition of scalar product between vectors where about q^2 solutions remain available after having fixed one vector in the product and the product value, while the conventional scalar product between scalars would

remain only a unique solution. That is another benefit in using vectors instead of scalars over $GF(q)$.

Suppose the cluster composed by the clusterhead n_{CH} and the sensor units n_i and n_j. In sec. 4.1 we stated that quantities into the definitions for the Local Key Component and the Transmitted Key Component in Sec. 4.1 f, i.e. $\overline{a} \in A$, $\overline{c} \in C$, $\overline{m} \in M$, and the scalar $b \in B$, can be freely chosen with compliancy to the (weak) constraints in Sec. 4.1 b, c. We can show that just adding only a further constraint in the selection of $\overline{a}_{CH}, \overline{a}_i, \overline{a}_j \in A$, a cluster-wise TAK, i.e. a TAK such that $TAK_{CH,i} = TAK_{CH,j}$ with $TAK_{i,j} \neq TAK_{CH,i}, TAK_{CH,j}$ can be generated.

Theorem 2 (Cluster-wise TAKS2 Generation). *Suppose the cluster composed by the clusterhead n_{CH} and the sensor units n_i and n_j. Given $\overline{c} \in C$, $\overline{m} \in M$, $b \in B$ and $\overline{a}_{CH}$, $\overline{a}_i$ and $\overline{a}_j$ compliant to the constraint*

$$\begin{cases} \overline{m} \cdot h(\overline{a}_j - \overline{a}_i) = 0 \\ \overline{m} \cdot (h(\overline{a}_j - \overline{a}_i) \times \overline{a}_{CH}) = 0 \end{cases}$$

for an arbitrary $h \in GF(q)$. Then $TAK_{CH,i} = TAK_{CH,j}$ is the cluster TAK with $TAK_{i,j} \neq TAK_{CH,i}, TAK_{CH,j}$. The same result applies to clusters of any size.

Proof. The proof is straightforward. Developing the complete expression for each pair-wise TAK given in Theorem 1, we get

$$\begin{cases} TAK_{CH,i} = \overline{k}_{lCH} \cdot \overline{k}_{ti} = b^{\overline{m} \cdot (\overline{a}_{CH} + \overline{a}_i + \overline{c})} \overline{m} \cdot (\overline{a}_i \times \overline{a}_{CH}) \\ TAK_{CH,j} = \overline{k}_{lCH} \cdot \overline{k}_{tj} = b^{\overline{m} \cdot (\overline{a}_{CH} + \overline{a}_j + \overline{c})} \overline{m} \cdot (\overline{a}_j \times \overline{a}_{CH}) \end{cases}$$

From the condition $TAK_{CH,i} = TAK_{CH,j}$ we get the constraints set

$$\begin{cases} \overline{m} \cdot (\overline{a}_j - \overline{a}_i) = 0 \\ \overline{m} \cdot ((\overline{a}_j - \overline{a}_i) \times \overline{a}_{CH}) = 0 \end{cases}$$

for which vector $(\overline{a}_j - \overline{a}_i)$ must be orthogonal to $\overline{m}$ and $\overline{a}_{CH}$ must lie on the plane identified by vectors $(\overline{a}_j - \overline{a}_i)$ and $\overline{m}$. Suppose to add a further member node, say node n_k, to cluster: the constraints set becomes

$$\begin{cases} \overline{m} \cdot (\overline{a}_j - \overline{a}_i) = 0 \\ \overline{m} \cdot ((\overline{a}_j - \overline{a}_i) \times \overline{a}_{CH}) = 0 \\ \overline{m} \cdot (\overline{a}_k - \overline{a}_j) = 0 \\ \overline{m} \cdot ((\overline{a}_k - \overline{a}_j) \times \overline{a}_{CH}) = 0 \end{cases}$$

where first and third equations enforce vectors $(\overline{a}_j - \overline{a}_i)$ and $(\overline{a}_k - \overline{a}_j)$ to lie on the same plane orthogonal to $\overline{m}$ while second and forth equations enforce $(\overline{a}_j - \overline{a}_i)$ and $(\overline{a}_k - \overline{a}_j)$ to be parallel, or $(\overline{a}_k - \overline{a}_j) = h(\overline{a}_j - \overline{a}_i)$ for an arbitrary $h \in GF(q)$, and $\overline{a}_{CH}$ to lie lie on the plane identified by vectors $\overline{m}$ and $h(\overline{a}_j - \overline{a}_i)$. Therefore the constraints set can be compactly written as

$$\begin{cases} \overline{m} \cdot h(\overline{a}_j - \overline{a}_i) = 0 \\ \overline{m} \cdot (h(\overline{a}_j - \overline{a}_i) \times \overline{a}_{CH}) = 0 \end{cases}$$

for an arbitrary $h \in GF(q)$. The condition

$$TAK_{i,j}, ..., TAK_{j,k} \neq TAK_{CH,i}, TAK_{CH,j}, ..., TAK_{CH,k}$$

can be shown as follows. Suppose the absurd case

$$TAK_{i,j}, ..., TAK_{j,k} = TAK_{CH,i}, TAK_{CH,j}, ..., TAK_{CH,k}$$

which gives

$$\begin{cases} \overline{m} \cdot h(\overline{a}_j - \overline{a}_i) = 0 \\ \overline{m} \cdot (h(\overline{a}_j - \overline{a}_i) \times \overline{a}_{CH}) = 0 \\ \overline{m} \cdot h'(\overline{a}_{CH} - \overline{a}_j) = 0 \\ \overline{m} \cdot (h'(\overline{a}_{CH} - \overline{a}_j) \times \overline{a}_i) = 0 \\ \overline{m} \cdot h''(\overline{a}_i - \overline{a}_{CH}) = 0 \\ \overline{m} \cdot (h''(\overline{a}_i - \overline{a}_{CH}) \times \overline{a}_j) = 0 \end{cases}$$

for arbitrary $h, h', h'' \in GF(q)$: first, third and fifth equations would enforce vectors $h(\overline{a}_j - \overline{a}_i)$,$h'(\overline{a}_{CH} - \overline{a}_j)$ and $h''(\overline{a}_i - \overline{a}_{CH})$ to lie on the same plane orthogonal to $\overline{m}$ while second, forth and sixth equations would enforce $\overline{a}_{CH}$ to be parallel to $h(\overline{a}_j - \overline{a}_i)$, $\overline{a}_i$ to be parallel to $h'(\overline{a}_{CH} - \overline{a}_j)$ and $\overline{a}_j$ to be parallel to $h''(\overline{a}_i - \overline{a}_{CH})$ or

$$\begin{cases} \overline{a}_{CH} = h(\overline{a}_j - \overline{a}_i) \\ \overline{a}_i = h'(\overline{a}_{CH} - \overline{a}_j) \\ \overline{a}_j = h''(\overline{a}_i - \overline{a}_{CH}) \end{cases}$$

which is not solvable for any $h, h', h'' \in GF(q)$. □

The interpretation of this result is that for any cluster can be associated a plane identified by vectors $\overline{m}$ and $h(\overline{a}_j - \overline{a}_i)$, with $\overline{a}_i$, $\overline{a}_j$ from a generic couple of member nodes, where $\overline{a}_{CH}$ must lie in. If a backup cluster-head is defined for the same cluster, say CH', then $\overline{a}_{CH'}$ must lie in the same plane as well.

4.2 Network Topology Authentication

As in [15], network topology authentication is still based on two main elements: a verification function $g(p, v)$ and a set of Topology Vectors, $T(i)$, corresponding to eligible neighbors nodes of n_i .
We have already defined $T(i) = \{\overline{t}_{\sigma(i)}\} = \{\overline{k}_{t\sigma(i)}\}$ as the set of $\sigma(i)$ Topology Vectors stored in node n_i. According to upgrades listed in Section 3, we directly set $\overline{t}_j \equiv \overline{k}_{tj}$. Let $g(p, v)$ be a function where p is a characteristic parameter of the entity to be authenticated (e.g. node n_j), or the prover, and v a characteristic parameter of a reference authentic value (e.g. node n_i), or the verifier.
Let $MAC(\cdot)$ be an cryptographic hash function [17], we define as verification function $g(p, v) = g(SS_j, SS_i) = MAC(SS_j) - MAC(SS_i)$. It is straightforward show that definition of $g(\cdot)$ is compliant to R4 and R5.

Theorem 3 (Network Topology Authentication). *In a node pair n_i and n_j, if $MAC(SS_j)$ computed by receiver n_j results equal to $MAC(SS_i)$ computed by transmitter n_i, then n_i is network topology authenticated by n_j.*

Proof. Node n_j computes $SS_j = \overline{k}_{lj} \cdot d$. Node n_i computes $SS_i = \alpha\overline{k}_{li} \cdot \overline{k}_{tj}$. If $g(SS_j, SS_i) = 0$, we have that $MAC(SS_j) = MAC(SS_i)$. Cryptographic hash function collision property [17] implies that $SS_j = SS_i$ or that $\overline{k}_{li}$, $\overline{k}_{lj}$, $\overline{k}_{ti}$ and $\overline{k}_{tj}$ are compliant to R2 and R3. Thus, n_i is network topology authenticated by n_j. □

5 Security Analysis

The following sections deal with these issues: in terms of computation of the entropy associated to TAKS2, complexity in breaking TAK generation algorithm and, lastly, robustness of TAKS2 at network level.

5.1 TAK Entropy

This section deals with the quantitative evaluation of TAK entropy. The following position is shown:

- $H(\overline{k}_{ti}) = H(\overline{k}_{tj}) = \log_2 q^3 = 3\log_2 q$
  This is straightforward to show because $\overline{k}_{ti}$ and $\overline{k}_{tj}$ are private data and moreover they are hidden by random moltiplication. Therefore, the uncertainty about $\overline{k}_{ti}$ and $\overline{k}_{tj}$ is maximum. Their entropy also is maximum (i.e. $3\log_2 q$).

Any operation on random randomizes the result, therefore we have that:

$$H(TAK) = H(\overline{k}_{li} \cdot \overline{k}_{tj}) = H(\overline{k}_{lj} \cdot \overline{k}_{ti}) = \log_2 q$$

or rather

$$\frac{H(TAK)}{\log_2 q} = 1bit/binit.$$

5.2 Security Level in a Single Node

This security level is calculated by evaluating the complexity to break the cryptographic key with a single node available. The security level in a single node of TAKS2 equals to security of TAKS [15]. In this case, also, it equals to the complexity in reverse engineering $\overline{m}$, $\overline{c}$, $\overline{a}$ and b from $\overline{k}_l$, $\overline{k}_t$ and the (public) expression of $f(.)$. The following system of equations show that the relationship between $\overline{k}_l$, $\overline{k}_t$ and $\overline{m}$, $\overline{c}$, $\overline{a}$ and b is not simply a discrete logarithm, which is one of most difficult problem in $GF(q)$ algebra [10], but becomes more complex due to $\overline{m}$ and $\overline{a}$ appearing as multiplying factors of the exponentiation and in the exponent.

$$\begin{cases} \overline{k}_l = \overline{a} b^{\overline{m} \cdot (\overline{a} + \overline{c})} \\ \overline{k}_t = (\overline{m} \times \overline{a}) b^{\overline{m} \cdot \overline{a}} \end{cases}$$

5.3 Security Level in the Network

This security level is calculated by evaluating the complexity to break the cryptographic key with all nodes in the network available. The *T-Security* concept is introduced.

Definition 1. *Given a network with N nodes, a cryptographic key is T-Secure if an attacker should capture* $T + 1 < N$ *nodes in the network to gain enough information to crack the key.*

The best case is when $T = N$, because in this case the cryptographic key never can be violated as there is no enough information shared in the network to do that. This result can be achieved if a share of the information needed to generate cryptographic keys is external to the network (i.e. residing in an external server). As proved for TAKS [15] also TAKS2 is *N-secure*.

6 Implementation Issues

The encryption scheme proposed in this paper belongs to a wider research project whose aim is to realize a middleware for secure WSN [13] [14]. The middleware will provide one or more encryption and decryption schemes and an *intrusion detection system* (such as [12]) integrating them in *Agilla* [2] an agent-based middleware developped for *TinyOS 1.x*. After its definition, we have implemented TAKS2 in TinyOS 1.x to facilitate its integration in Agilla. Hereafter, we introduce concepts of TinyOS needed to understand TAKS2 implementation and then we discuss about implementation strategy.

6.1 TinyOS Programming

A TinyOS program consists of a minimal *scheduler* and a graph of *components* [7]. The scheduler can be seen as a service provided by the operating system not directly used by programmers. Then we focus on components.
A component is a self-conteined module of the TinyOS program: it can *use* services *provided* by other components and *provide* services that other components can *use*. These services are grouped in *interfaces*. These interfaces are the only point of access to the component and are bidirectional. An interface declares a set of functions called *commands* that the interface provider must implement and another set of functions called *events* that the interface user must implement. A single component may use or provide multiple interfaces and multiple instances of the same interface. For example, we can consider the TinyOS component which deals with radio transmission and reception: we expect one or more interfaces that define a command to send messages and an event to handle their reception.
Moreover, commands and events handlers are not atomic so, for long elaborations, TinyOS provides *tasks*. Tasks are atomic to each other. They are scheduled by the application scheduler with FIFO policy and can be preempted by events

and commands. For this reason the design and implementation of the events and commands handlers of a TinyOS component typically provide storing of their context (i.e. actual parameters of the function) and consequently posting of the elaboration to a task.

6.2 The SecureComm Component

In this section, we present *SecureComm*, the component implements TAKS2. First of all, it is described the SecureComm component itself (i.e. interfaces provided and used) and then we propose a pseudo-code version very similar to the real code.

SecureComm component

```
SecureComm
{
  provides
  {
    interface StdControl;
    interface SendMsg;
    interface ReceiveMsg;
  }
  uses
  {
    interface SendMsg;
    interface ReceiveMsg;
    interface StdControl;
    interface Random;
    interface MAC;
    interface BlockCipherMode;
  }
}
```

Generally, software design of any encryption scheme must be done so that using the scheme is completely transparent to the user. To satisfy this condition the security layer must provide the same interface of underlying layer. TAKS2 is implemented on physical layer provided by TinyOS and so it offers the same interface.
GenericComm is the TinyOS component that users exploit to interact with the physical layer. It is possible to send and receive messages through the *SendMsg* and *ReceiveMsg* interfaces provided by the component. Then, we have implemented the SecureComm component so that it provides SendMsg and ReceiveMsg interfaces. Of course, SecureComm also uses SendMsg and ReceiveMsg of GenericComm to be able to send and receive radio packets. SecureComm also provides *StdControl* that is the TinyOS standard interface to initialize and de-initialize the component itself. Finally, it uses components offering *Random*, *MAC* and *BlockCipherMode* interfaces to generate a random, to compute

MAC tag and to encrypt/decrypt messages. Actually these components are *RandomLFSR*, *CBCMAC* and *CBCModeM* respectively.

SecureComm component pseudo-code

```
/*Command called at every "send a message" request*/
command send(addr, length, plain_txt) {
  if(length < MAX_LENGTH) {
    if(busy == FALSE) {
      busy = TRUE;
      save_info(addr, length, plain_txt);
      post send_message();
    }
  }
}

/*Task that handles the logic to send a message*/
task send_message() {
  alpha = rand();
  tak = get_tak(addr);
  SS = multiply(alpha,tak);
  c = encrypt(SS, plain_txt, MAX_LENGTH);
  tau = mac(SS,c);
  d = multiply(-alpha,kt);
  GenericComm.send(BROADCAST, MAX_LENGTH,c|d|tau);
}

/*Function that initializes and encrypts the plain text*/
encrypt(SS, plain_text, length ) {
  CBCModeM.init(SS);
  return CBCModeM.encrypt(plain_text,  length);
}

/*Function that initializes and computes the MAC tag*/
mac(SS, text) {
  CBCMAC.init(SS);
  return CBCMAC.MAC(text);
}

/*Event called when send is done*/
event sendDone(addr, length, plain_txt) {
  busy = FALSE;
}

/*Event called when a radio packet is received*/
event GenericComm.receive(rcv) {
  if(busy == FALSE) {
```

```
    busy = TRUE;
    save_info(rcv);
    post receive_message();
  }
}

/*Task that handles the logic to receive a message*/
task void receive_message() {
  SS = inner_product(rcv->d, kl);
  tau = mac(SS,rcv->c,MAX_LENGTH);
  if(tau == rcv->tau) {
    plain_txt = decrypt(SS, rcv->c, MAX_LENGTH);
    busy = FALSE;
    signal UpperComponent.receive(plain_txt);
  }
}

/*Function that initializes and decrypts the cipher text*/
dencrypt(SS, plain_text, length) {
  CBCModeM.init(SS);
  return CBCModeM.dencrypt(plain_text, length);
}
```

Any component that needs to send a radio message uses the *send*() command of GenericComm. We duplicate this command in SecureComm to save context information and to delegate further elaboration to *send_message*() task.
This task generates a random α and it computes the TAK knowing the address of destination node that is used to access Local Planned Topology table implemented on node. Then it can generate the secret share SS to encrypt the plain text (producing c) and get the message authentication code (τ). So, the task can send the whole packet by using the send() command of GenericComm. It is worth noting that encryption and MAC tagging are done respectively by *encrypt*() and *mac*() functions. These functions currently use components CBCModeM and CBCMAC from *TinySec* library although the component is flexible to work with different ones. In fact, it is sufficient to change these components with other ones offering *BlockCipherMode* and *MAC* interfaces respectively. The discussion on encryption is concluded observing that *multiply*() and *inner_product()* have been implemented to work on operands of 128 bit as well as the other functions and the entire scheme.
Message reception is very similar to transmission. To handle received message is needed to implement a new event handler *wired* to *receive*() handler of GenericComm (we omit the wiring operation in the pseudo-code to avoid too much details). Such an handler saves information and delegates the elaboration to *receive_message*() task. Such a task computes the secret share based on deciphering information (d) and Local Key Component. Then, the task can authenticate the message accepting it if authentic or discarding it otherwise. It

is important highlight that every message is broadcast delivered. So the above procedure is executed by every node in the range of the transmitter, but only the actual destination node is able to correctly decrypt sent packet. Finally also for deciphering, the component SecureComm is able to work with different components as long as they are compliant with components used in encryption.

6.3 Cost Analysis and Execution Time

In this section, we describe the cost analysis of proposed scheme and its execution time. This analysis aims to evaluate the complexity of the scheme regardless of the encryption and decryption algorithm and message authentication coder (since they are always needed and could be freely selected by the network planner). So, we do not care of encrypt(), decrypt() and mac() spatial and computational complexity. In Table 1 we report computational complexity of other functions. Let n and $\sigma(i)$ be the key size in bytes and the cardinality of the set of eligible neighbors respectively:

- to generate a n-bytes random, we need to generate n random of 8 bit, so $rand()$ costs $O(n)$;
- to add two n-bytes number, we need to do n addition of 8 bit data, so addition costs $O(n)$;
- to multiply two n-bytes number, we need to do $\frac{n(n+1)}{2}$ moltiplication and n addition of 8 bit data, so addition costs $O(n^2)$;
- to do inner product of two vector of 3 n-bytes components, we need to do 3 n-bytes moltiplication and 2 n-bytes addition so inner product costs $O(n^2)$;
- to get TAK from destination address we need to find Topology Vector of destination node (that costs $\sigma(i)$) and to do inner product with proper Local Key Component. So, to get TAK from destination address costs $O(n^2+\sigma(i))$.

Table 1. Computational complexity of TAKS2 functions

Function	t(n)	$O_t(n)$
$rand()$	n	$O(n)$
$get\_tak()$	$\sigma(i) + 3n + 3\frac{n(n+1)}{2}$	$O(n^2 + \sigma(i))$
$multiply()$	$\frac{n(n+1)}{2} + n$	$O(n^2)$
$inner\_product()$	$3\frac{n(n+1)}{2} + 2n$	$O(n^2)$

Since send_message() task is a serialized call of rand(), get_tak() and multiply() we can affirm that computational complexity of TAKS2 encryption is $O(n^2+\sigma(i)) \simeq O(n^2)$. Similarly, since receive_message() task equals to complexity of inner_product() we can affirm that computational complexity of TAKS2 decryption is $O(n^2)$. This result is not a problem since we have good security

properties with 128 bit keys.
The spatial complexity, due to the creation of temporary structures for mathematical calculations, is $O(n+\sigma(i))$. So, the spatial complexity is not a constraint for the execution of the scheme.
To calculate the execution time of the encryption and decryption scheme we have used an *enriched version* of SecureComm component to send periodically test packets. To get the execution time we have made SecureComm able to get system time of the node through the *SysTimeC* TinyOS component. Furthermore, enriched version of SecureComm is also able to send arbitrary packets via UART, so that we can get information by the node during its normal behavior. Sampling system time in appropriate point in the code and sending this information via UART we have been able to calculate execution time of various operations carried out by node. With this solution, send and receive tasks are $15.42ms$ and $8.34ms$ long respectively with encryption, decryption and MAC calculation $1.83ms$, $1.97ms$ and $1.87ms$ long respectively.
For that concern memory occupancy considering $\sigma(i) = 5$ meaningful (as in [16]), nesC compiler reports 1375 bytes as occupancy in RAM.

6.4 Validation

The proposed scheme has been validated by means of a testbed designed to show that:

T1. two mutually eligible nodes are able to communicate;
T2. only the actual destination node is able to correctly decrypt a packet.

To prove test T1 and T2 we have deployed a network of 3 nodes. In this network, each node is within range of other ones. Fig.2 shows the planned network topol-

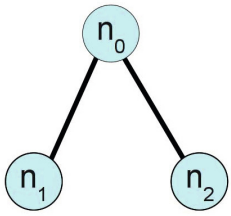

Fig. 2. Testbed Planned Network Topology

ogy. Starting from planned network topology depicted in Fig.2. we randomly produce ($\overline{a}_0$,$\overline{a}_1$,$\overline{a}_2$,b,$\overline{c}$,$\overline{m}$) and we calculate ($\overline{k}_{l0}, \overline{k}_{t0}$), ($\overline{k}_{l1}, \overline{k}_{t1}$), ($\overline{k}_{l2}, \overline{k}_{t2}$) as described in Section 4.
Therefore, we get the following for LCD:

$$LCD_0 = \{\overline{k}_{l0}, \overline{k}_{t0}, T(0) = \{\overline{k}_{t1}, \overline{k}_{t2}\}\}$$
$$LCD_1 = \{\overline{k}_{l1}, \overline{k}_{t1}, T(1) = \{\overline{k}_{t0}\}\}$$
$$LCD_2 = \{\overline{k}_{l2}, \overline{k}_{t2}, T(2) = \{\overline{k}_{t0}\}\}$$

The design of the testbed is based on typical features of monitoring wireless sensor networks. In these networks, a set of nodes send data to a special node called *base station*. In our network, node n_0 behaves as base station while other ones are dedicated to data acquisition.
To validate the scheme, we have implemented an application on SecureComm so that:

- n_1 and n_2 send a test message (rather than sensor data) to base station at predefined rate;
- each node notify message reception toggling a led;
- each node notify *authenticated* message reception toggling a led and sending the message via UART interface.

Previous application facilities are chosen so that test T1 and T2 can be easily proved. The former by verifing that at each transmission base station send the expected test message on UART. The latter by verifing that base station is the only node to toggle the authentication led although other node also receives the message. This testbed has been run both on *TOSSIM* (a simulator for TinyOS networks) and on a real network of MICAz nodes.

7 Conclusions and Future Works

This paper has proposed a novel scheme to generate topology authenticated keys in Wireless Sensor Networks. Its effectiveness has been proved both formally and experimentally. In particular, its robustness has been proved by showing that entropy of the keys is high and the scheme is N-secure (i.e. the attacker should capture N nodes in the network to gain enough information to crack the key). Moreover, since key size in symmetric schemes is quite limited, the computational complexity of the scheme ($O(n^2)$) is a very result with respect to its robustness. The described work belongs to a wider research project whose aim is to develop a secure WSN middleware. Such a middleware will provide a encryption and decryption scheme associated with an intrusion detection system. Future works foresee defining and implementing intrusion detection techniques to form a complete secure WSN middleware.

Acknowledgment. The research leading to these results has received funding from the European Union Seventh Framework Programme [FP7/2007-2013] under grant agreements n. 257462 HYCON2 Network of excellence and n. 240555 ERC SG VISION. Moreover, it has been motivated and supported by the ESF-COST Action IntelliCIS (Prof. Fortunato Santucci is participating to this Action). The development of the middleware platform also fits in the frame of the Projects Ricostruire and SMILING supported by the Ministry of Economic Development to enhance technology transfer in the RIDITT framework.

References

1. Abbasi, A.A., Younis, M.: A Survey on Clustering Algorithms for Wireless Sensor Networks. Computer Communications 30 (2007)
2. Agilla Home Page, `http://mobilab.wustl.edu/projects/agilla/`
3. Blom, R.: An optimal class of symmetric key generation systems. Eurocrypt 84 (1985)
4. Camtepe, S.A., Yener, B.: Key distribution mechanisms for wireless sensor networks: a survey. Techical Report TR-05-07, Troy (2005)
5. Chan, H., Perrig, A., Song, D.: Random key predistribution schemes for sensor networks. In: IEEE Symposium on Research in Security and Privacy (2003)
6. Hankerson, D., Menezes, A., Vanstone, S.: Guide to Elliptic Curve Cryptography. Springer, New York (2004) ISBN 0-387-95273-X
7. Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., Pister, K.: System Architecture Directions for Networked Sensors. In: Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX), Cambridge, MA, USA, pp. 93–104 (November 2000)
8. Lai, B., Kim, S., Verbauwhede, I.: Scalable session key construction protocol for wireless sensor networks. In: IEEE Workshop on Large Scale RealTime and Embedded Systems, LARTES (2002)
9. Liu, D., Ning, P.: Establishing pairwise keys in distributed sensor networks. In: 10th ACM Conference on Computer and Communications Security, CCS 2003 (2003)
10. Menezes, A.J., Van Oorschot, P., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press (1996)
11. Vlajic, N., Xia, D.: Wireless Sensor Networks: To Cluster or Not To Cluster? In: Proceedings of the 2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2006), Buffalo (2006)
12. Pugliese, M., Giani, A., Santucci, F.: A Weak Process Approach to Anomaly Detection in Wireless Sensor Networks. In: First International Workshop on Sensor Networks, SN 2008 (2008)
13. Pugliese, M., Pomante, L., Santucci, F.: Agent-based Scalable Design of a Cross-Layer Security Framework for Wireless Sensor Networks Monitoring Applications. In: Proceedings of the International Workshop on Scalable Ad Hoc and Sensor Networks (SASN 2009), Saint Petersburg (2009)
14. Pugliese, M., Pomante, L., Santucci, F.: Secure Platform over Wireless Sensor Networks. INTECH Publishers (2012) ISBN 978-953-51-0218-2
15. Pugliese, M., Santucci, F.: Pair-wise Network Topology Authenticated Hybrid Cryptographic Keys for Wireless Sensor Networks using Vector Algebra. In: 4th IEEE International Workshop on Wireless Sensor Networks Security (WSNS 2008), Atlanta (2008)
16. Pugliese, M., Pomante, L., Santucci, F.: Topology Optimization and Network Deployment Algorithm in WSNs for Mobile Agent-based Applications. In: 4th European Modelling Symposium, EMS 2010 (2010)
17. Rogaway, P., Shrimpton, T.: Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision-Resistance. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 371–388. Springer, Heidelberg (2004)
18. TinyOS Home Page, `http://www.tinyos.net`
19. Wander, A.S., Gura, N., Eberle, H., Gupta, V., Shantz, S.C.: Energy analysis of public-key cryptography for wireless sensor networks. In: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications (PERCOM 2005), Washington, pp. 324–328 (2005)