Journal Title
XX(X):1–15

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

# Development of an extended *Topology-based* Lightweight Cryptographic Scheme for IEEE 802.15.4 Wireless Sensor Networks

**Walter Tiberti[1], Federica Caruso[1], Luigi Pomante[1], Marco Pugliese[1], Marco Santic[1] and Fortunato Santucci[1]**

**Abstract**

Among the classes of *Wireless Personal Area Networks* (WPANs), a Wireless Sensor Network (WSN) typically refers to a versatile and densely distributed sensing platform that enables the support of a wide variety of application domains. Among the various technical challenges addressed by more than one decade of research in WSNs, security across wireless links is by far one of the most critical ones and relates to the need of guaranteeing reliability and trustiness of the collected data. This paper deals with the cryptographic aspects involved in securing WSNs, in terms of confidentiality and authentication. In particular, moving from some results previously achieved in our research activity, this paper extends a cryptography scheme in order to better comply with the security requirements that arise from real-world WSN installations. The proposed scheme, called TAKS2, takes advantage of Hybrid Cryptography to provide security in the presence of resource-constrained sensor nodes by using topology-authenticated keys to provide increased robustness to the scheme itself. The proposed extensions provide full practical support to star-topology WSNs and the paper presents also some experimental results obtained by implementing the scheme on two different WSN platforms available for protocol stacks compliant with the IEEE 802.15.4 standard.

**Keywords**

WSN security, cryptographic scheme, hybrid cryptography, topology authenticated key

## Introduction

A Wireless Sensor Network (WSN) is a network class where small, battery-powered sensor-equipped nodes (called *sensor nodes* or also *motes*) communicate by using low power, short range, and low data rate wireless technologies. WSNs are designed to provide high flexibility, high measurement redundancy and potential wide sensor area coverage with small costs. A typical mote consists of a *micro-controller unit* and a *radio transceiver* powered by a battery pack. Computational power and available memory on a mote are normally highly constrained. As a consequence, software running on motes has to be as lightweight as possible. Moreover, such limitations led software developers to adopt low-level programming languages (and, sometimes, also assembly) to minimize both WSN software execution time and memory occupation. However, projects such as TinyOS [23], ContikiOS [24] and Riot-OS [25] propose lightweight low-level software frameworks (assimilable to *Operating Systems*, hence the *OS* suffix) to enhance and make easier the software development process for WSNs. An important characteristic of these operating systems is the *ready-to-use* software primitives and components for, e.g., resource management, communication protocols and stacks, as well as the compatibility with different hardware platforms, so that developers can focus on writing the high-level logic of WSN applications, leaving the hardware heterogeneity issues to the OS code.

Depending on the context in which a WSN is deployed, the need for *security* in motes communications may arise. As any other network, *confidentiality*, *integrity* and *authentication* can be enforced by means of cryptography, error detection/correction and message authentication techniques. However, due to the hard constraints on computing power, memory and available energy in WSN motes, standard state-of-the-art mechanisms and implementations of security-related algorithms are not always possible. In particular, considering confidentiality and authentications, the accepted techniques require the usage of *public-key* (i.e., asymmetric) cryptography, which implementations require an amount of memory (e.g., keys, temporary data, look-up tables) and computing power (e.g., modular-arithmetic, random number generation, factorization, exponentiation, operations on points) which can represent an issue when combined with the actual applications on the motes. On the other hand, techniques involving symmetric cryptography, which are inexpensive and fast, require a key management mechanism to provide each mote the right key to use.

[1]University of L'Aquila, Italy

**Corresponding author:**
Walter Tiberti, University of L'Aquila, L'Aquila, Italy
Email: walter.tiberti@univaq.it

In order to overcome the described issues, a modern and effective approach is represented by the *Hybrid Cryptography*. Such techniques mix the public-key and symmetric cryptography in order to overcome the limitations of both approaches.

## Background and motivations

Providing security primitives in traditional networks often implies the use of asymmetric cryptography mechanisms. This strategy guarantees an high degree of security, which is sustainable thanks to the ever increasing amount of available resources in terms of computation, memory and energy. In fact, the robustness of asymmetric cryptography algorithms is highly dependent on the size of the keys that in turn affects the algorithms complexity and the performance of their implementations. In this sense, the *Elliptic Curve Cryptography* (ECC) [4] is able to provide a very high degree of security with keys shorter than the ones used in other state-of-art public-key algorithms (e.g., RSA [20], ElGamal [21], etc.). On the other hand, while symmetric cryptography schemes can grant equivalent security and better performances, they require a mechanism for securely exchanging, updating and revoking the cryptographic keys among the parties. However, when the computation resources, the memory and the available energy become scarce, the complexity associated to asymmetric cryptography make it unfeasible in practice.

Although *modified* asymmetric cryptography schemes have been discussed [17] [18], in this context, the role of symmetric cryptography coupled to a *key management* technique becomes relevant again. In fact, key management is a fundamental problem in symmetric cryptography and it has a wide coverage in literature [2].

In the scheme proposed in this paper, with respect to the existing key management solutions (see sec. 1.2), keys are not fully-distributed to each node of the network. Instead, they are the result of a dynamic generation phase based on partial key components (pre-distributed and locally stored in each node) as inputs, as in [8]. By combining components with low-complexity operations, nodes are able to compute the symmetric decrypt/encrypt key without any prior setup/negotiation.

More in detail, the proposed scheme Topology Authenticated Key Scheme (TAKS) represents a quite extension of ephemeral Diffie-Helman-like cryptographic protocol providing security over a resource constrained network (typically ad-hoc networks, e.g. sensor networks for monitoring services) to establish both point-to-point and point-to-multipoint secure links among nodes. Such a scheme, called TAKS2, uses Hybrid Cryptography and only partial components of symmetric keys are pre-distributed (and not the whole keys). It is an evolution of the original TAKS, that was earlier presented in [8] and in [36]. The authentication mechanism of the scheme is based on nodes topology rather than on nodes identity, due to the limited lifetime of nodes in a resource-constrained network. Indeed, while nodes in infrastructured networks can rely on an external power supply and on a stable planned maintenance service with human intervention, the nodes in ad-hoc networks can rely only on their own battery or some other energy harvesting mechanism and maintenance services are usually remotely performed without human intervention. When an off-duty node is replaced with a new node, the new node identity enters in the network, but node topology remains unchanged and the authentication mechanism does not need any updating. Security features of TAKS2 include confidentiality (data encryption), data integrity and sender authentication (signature). In TAKS2 the shared secret is a symmetric key generated by each party involved in the communication session upon a successful authentication process: each party verifies if the other party belongs to its authenticated network. The assignment parameter to each node is carried out by an external Certification Authority (CA), next the scheme parameters are pre-loaded into the nodes. Vector space rather than scalars over Galois fields has been introduced to allow the setup of truly scalable multicast communication sessions, hence without setting up multiple unicast sessions: the added dimensionality respect to scalars introduces a further degree of freedom for CA in the procedure of scheme parameters computation and assignment. From an engineering point of view multicast sessions provide a relevant feature, especially for clustered networks, where time synchronization in data transmission is required as well as lighter memory storage: multicast sessions avoid traffic flooding, hence wasting of energy, when maintenance services are activated on such clustered networks (e.g. the updating of some configuration parameters in a specific portion of the network). With respect to classical solutions of key-assignment schemes on network, the scheme proposed in this paper does not rely on pre-distribution of the complete keys but of a component of them: this is a nice property from a robustness point of view because the shared secret in ECTAKS is function of both sender and receiver private key components while in odinary ephemeral Diffie-Hellman-like schemes typically the shared secret is function of the complete sender whole private key and the receiver whole public key.

Moreover, the paper reports the details of the implementation of the proposed scheme both in *nesC/TinyOS 2.x*, an operating system for a variety of families of sensor nodes [23], and in bare-metal *C* language (supported by Atmel legacy libraries [33]). Finally, the paper describes the experimental results obtained by means of real deployments on two hardware platforms: a clone of the famous Texas Instruments *TelosB* sensor node and a legacy sensor node provided by Atmel, both compliant with the IEEE 802.15.4 standard protocol. To resume, the main contributions of this paper are:

- TAKS2 SW design has been improved to consider also star topologies other than point-to-point ones (Section 4.1);
- TAKS2 has been implemented on top of the MAC layer of two different IEEE 802.15.4-enabled WSN hardware platforms (Section 4.2);
- a *testbed* application has been developed to simulate a real-world monitoring scenario (Sections 5.1 and 5.2);
- this testbed has been used to validate the correctness of the scheme and to evaluate its performances in terms of computation time and memory occupation (Sections 5.3 and 5.4).

### Comparison with state-of-the-art solutions

Symmetric cryptography schemes, such as *Advanced Encryption Standard* (AES, [19]), have usually very high performance and limited footprint in memory. Moreover, to improve performance, various micro-controllers and radio transceivers provide symmetric cryptography primitives directly as hardware accelerators. However, in symmetric schemes, the symmetric key needs to be known and available by every node before sending or receiving messages.

This requirement leads to the *Key Distribution* and *Key management* issues frequently addressed in the literature. In fact, there are solutions based on *key pre-distribution* (e.g. [13]), which consist of the deterministic pre-distribution of keys. The trivial security solution consists in distributing a key for each pair of nodes, and, as extreme (and less secure) case, the same symmetric key for each node of the network. A derivation of the previous mechanism is the *Random pair-wise key schemes* which consist of storing only a subset of all possible keys in each node [3]. Nodes that wish to communicate have to negotiate a key with their peers, by selecting a random key from their subset. A simplification is possible if nodes locations are known, by providing to each node only the keys for the actual neighbours [6]. In cluster-based networks, it is possible to adopt a *cluster pre-distribution scheme*. Clusters use different keys [5], while, within a cluster, the same key is used. Some mechanisms, e.g., the *master key pre-distribution* [2] define a *master key* which is used (combined with some previously exchanged *nonce* values) by nodes to re-create the symmetric key. Some schemes [12] use different paradigms to distribute the keys, such as the use of *mobile-agents* capable of moving inside and distribute the cluster key dynamically and with reduced energy consumption. Alternative schemes use the *key-matrix based dynamic key generation* [1], the *Identity-based Encryption* [10] and the *Threshold Cryptography* [11] to manage and distribute keys while reducing the energy consumption.

In this context, while TAKS2 does not achieve the same performance of a pure symmetric scheme, it resolves the secure key distribution problem with a small impact on performance. On the other side, public-key schemes (e.g., RSA, ECC) solve the key distribution problem and provide a higher degree of security at the cost of increased impact on performance and memory. For example, in a typical WSN node with a 8-bit micro-controller, the modular arithmetic operations, such as addition and multiplication, have to be implemented by means of algorithms which operate byte-wise on the numbers/points, introducing a non-negligible impact on performance and memory. In TAKS2, instead, the complexity and impact of such operations are moved to the key components generation: once the key components are available to nodes, the operation to combine them so obtaining the symmetric key (i.e., $TAK(\cdot)$) has a very lightweight impact on performance. This advantage also affects the communication throughput, since faster encryption/decryption operations result in faster communications.

In comparison with other hybrid-cryptography schemes (e.g., [39][40][41][42]) TAKS is suitable for real-world WSN motes, has lower average overhead both in timing performance and memory occupation and provides the possibility of choosing different approaches for symmetric encryption/decryption and for authentication of received messages. In Table 1, we present a brief comparison of common general cryptographic solutions and TAKS2, while in Table 2 a comparison with other state-of-the-art hybrid cryptographic schemes for WSN is shown. TAKS supports all the hardware platforms supported by the TinyOS [23] operating system, whereas in most of the state-of-the-art hybrid cryptography schemes the OS/hardware support is not clear or limited to software simulations. The performance overhead (apart from the networking stack overheads) is comparable or better than the other schemes with a memory occupation which is beaten only by adopting external hardware modules (as in [**?** ]). TAKS provides its own lightweight key exchange protocol (described in [36] and reported in this paper), symmetric key encryption with no restriction on the key size or the cipher used and an authentication primitive based on the *Message Authentication Code* (MAC) check. Other schemes, instead, have only a single (e.g., [40][41]) or fewer cryptographic capabilities.

### Paper Organization

The next sections of the paper are organized as follows. In Section 2, original *TAKS* scheme is resumed along with the enhancements of its second version, i.e., *TAKS2*. Section 3 covers the security analysis of TAKS2. Section 4 describes the theoretical computational and spatial complexity of TAKS2. In Section 5, design and implementation issues over two different WSN platforms are discussed while, in Section 6, the verification and the performance evaluation processes, with related experimental results, are described in detail. Finally, Section 7 concludes the paper with some comments and planned future works.

## Topology-Authenticated Key Scheme

TAKS (*Topology-Authenticated Key Scheme*) is a cryptographic scheme to provide passive security at link layer. TAKS is an hybrid deterministic KEP (*Key Establishment Protocol*) to establish both pair-wise and cluster-wise secure links. Its security functions include confidentiality (data encryption / decryption), data integrity and sender authentication (signature). The symmetric key is generated by the parties on a successful authentication process (*TAKS Authentication*) based on the verification that the involved parties belong to a predefined *Planned Network Topology* (PNT). We define qualitatively a PNT to be the network topology planned by the network planner in order to fulfill given requirements. In this context, the *eligible neighbour nodes* are defined as the nodes authorized to communicate with a target node. A PNT represents the network topology as it is expected to be at service runtime: therefore the PNT can be considered as the *Authenticated Network Topology* (ANT) for that portion of network.

Each node pre-stores a set of TAKS security parameters (denoted as *Local Configuration Data*, LCD) namely the *Local Key Component* (LKC), a set of *Transmitted Key Components* (TKCs) and and a set of *Topology Vectors* (TVs), each TV associated to a node belonging to the ANT. LKC takes the role of private key of the node while

**Table 1.** Comparison of TAKS2 against other solutions

| | RSA | ElGamal | ECIES | AES | **TAKS2** |
|---|---|---|---|---|---|
| Type | Public-Key | Public-Key | Public-Key | Symmetric | **Hybrid** |
| Key bit-length order | Thousands | Thousands | Hundreds | Hundreds | Hundreds |
| WSN Performance overhead | High | High | High | Low | Low |
| WSN Memory overhead | Moderate | Moderate | Moderate to High | Low | Low |
| Encryption + Authentication | Separated (RSA sign.) | Separated (DSA) | Separated (ECDSA) | Yes (CCM/GCM modes) | **Yes** |
| Secure against Node capture | Yes | Yes | Yes | No | **Yes** |

| Scheme | Supported Platform | Perf. Overhead TX / RX | RAM / ROM Overhead | Key Exchange | Symmetric Enc/Decryption | Authentication |
|---|---|---|---|---|---|---|
| **SCUR** [43] | TinyOS | ? | ? | Pre-distributed | Rabbit | None |
| **SecFleck** [44] | Hardware-module | ~800ms / ~800ms | 1082 B / 52 B | RSA-based | XTEA | RSA-based |
| **Tongxu et Al** [39] | ? | ~1000ms, ~1000ms | ? | ? | AES/ECC | MAC-based |
| **EHKM** [40] | Simulations | ? | N/A, ~1500 B | ECDH | None | None |
| **Attiya et Al** [41] | TOSSIM+MICA2 | ~26 ms (only key mgn) | ~34KB, ~2.7 KB | Cluster-based | None | None |
| **Manjunath et Al** [42] | Simulations | ~8s | ? | ECDH | ? | Certificate-based |
| **TAKS v2** (this work) | **TinyOS** | **~30ms, ~15ms** | **~10 KB, ~700 B** | **TAKS** | **Any, Selectable** | **MAC-Based** |

**Table 2.** TAKS2 comparison with other state-of-art hybrid-cryptography schemes. The table compares different features: the supported operating systems, the overhead (i.e., additional delay) introduced in communications, the amount of additional memory required to support the scheme, the adopted key exchange protocol (if any), the symmetric encryption and decryption protocols (if any) and the authentication protocol (if any). The question mark ('?') denotes that no information about the referenced feature has been found.

TKCs and TVs take the role of public key of the node / cluster of nodes. An (external) *Certification Authority* binds LKC to the node ID and TKCs / TVs to the nodes IDs (or Cluster IDs) associated to the ANT. Therefore, the membership of a node to a specific ANT is certified by the assignment of the associated TV. Any node in a specific ANT is an *Authenticated Node*. A TAK is generated if the involved parties belong to the same ANT (hence the attribute of "Topology-Authenticated Key"). The PNT is implemented as a distributed data structure, in which the information on the available communication links are stored for each node in the network. In particular, each node portion of the PNT contains (at minimum) the subset of TKCs and TVs relative to each one available to the node. Additional information could be stored if required by the TAKS implementation. TAKS2 is the enhanced version [36]: both represent a *Hybrid Cryptography* scheme, since they do not rely explicitly on asymmetric or symmetric keys to secure the WSN data transmissions. In literature, there exist other hybrid cryptography schemes (e.g. [14] [15] [16]) but TAKS2 is the only scheme proving WSNs with a secure encryption mechanism along a *Topology-based* authentication with minimal performance overhead and memory footprint.

## Description of the cryptographic scheme

Before describing the inner mechanisms of the cryptographic scheme, we present the security level definitions that will be adopted in the rest of the paper: [36]

- *public*: general public information, accessible by anyone (attackers included);
- *restricted*: network-level private information (accessible by nodes in the network);
- *private*: node-level private information;
- *secret*: planner-level private information.

As a requirement of the scheme, a data structure called *Local Configuration Data* (LCD) has to be stored inside the WSN nodes. The LCD contains the definition of scheme parameters, including the key partial components and topology information. Topology definition can be physical (e.g., with geographical position of nodes) or logical (i.e, in terms of which node pairs can communicate) depending if the scheme is applied on physical/MAC layer or upper layers respectively. LCD includes:

- the *Local Key Component* (*LKC*);
- the *Transmit Key Component* (*TKC*);
- and the *Planned Network Topology* (*PNT*).

LKC is a component that is kept secret and stored in the target node, while TKC is a public component that is distributed to every node enabled to communicate with the target node. PNT is a set of *Topology Vectors* (*TV*s) used to describe the relationship (one-to-one or one-to-many) among eligible neighbour nodes for pair-wise and cluster-wise link configuration respectively. The security of the 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.

In the following paragraph we introduce the scheme (details can be found in [8][36]).

Choose a large prime number $p$ such that $p \gg N$ where $N$ is the total number of nodes in the WSN. Let $U$ be a tri-dimensional vector space over $GF(p)$ with $\overline{u} \in U$ as generic vector. $\overline{u}$ is represented by $(u_x, u_y, u_z)$ with $u_x, u_y, u_z \in GF(p)$.

Let be a function, called $TAK(\cdot,\cdot)$ with the following characteristics:

**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 $\alpha \in GF(p)$;

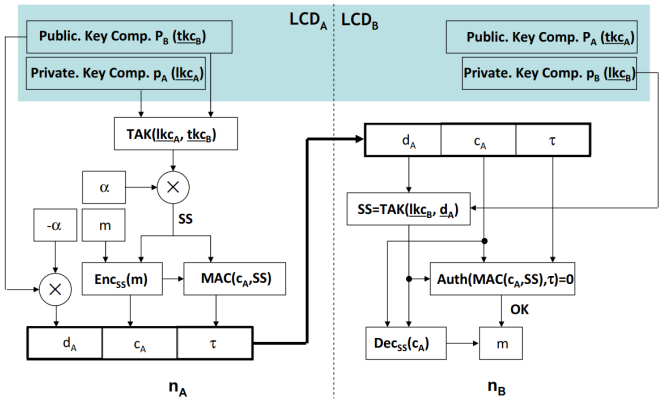


**Figure 1.** TAKS2 scheme description.

The $TAK(\cdot,\cdot)$ function is used to generate the Topology-Authenticated Keys from the pre-distributed components.

Let $g(\cdot,\cdot)$ 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 $p, v$.

The $g(\cdot,\cdot)$ function is used to verify the authenticity of every incoming message.

In Figure 1 a representation of the proposed scheme is shown.

Given the set of eligible neighbours nodes of node $n_i$ (the set $\sigma(i)$, the PNT of node $n_i$ is defined to be the set of topology vectors $TV(i) = \{TKC_{\sigma(i)}\}$. Along the PNT, each node stores also its $TKC_i$ and $LKC_i$.

In every communication, a *nonce* value $\alpha \in GF(p)$ has to be generated. If the node $n_i$ wants to communicate with $n_j$, it builds a message payload containing:

- the result of the symmetric key encryption of the message, the *cipher text* ($c$), using $SS = \alpha TAK(LKC_i, TKC_j)$ as symmetric key;
- the *key reconstruction information* (*KRI, d*), where $d \in U$ and $d = -\alpha TKC_i$;
- a *message authentication code* ($\tau$) (denoted as $MAC(\cdot)$) and computed using any secure *cryptographic keyed hash function* with $\alpha TAK(LKC_i, TKC_j)$ as key.

As shown in Figure 1, *alpha* and all the values generated from it (e.g., $SS$, $d$, etc.) are generated for each communication and never stored.

Upon the reception of a message, the node $n_j$ calculates back the symmetric key to correctly decrypt the content of the message. The key is computed with $TAK(LKC_j, d)$ since:

$$\alpha TAK(LKC_i, TKC_j) = \alpha TAK(LKC_j, -TKC_i)$$
$$= TAK(LKC_j, -\alpha TKC_i)$$
$$= TAK(LKC_j, d)$$

The message authenticity function $g(\cdot,\cdot)$ has to return zero when original encryption key and computed decryption key are identical. As shown in Figure 1, this computation uses $\tau$ to verify whether this condition is met.

Starting from the first version, TAKS has evolved into TAKS2 [8][36]. This second version has the following main features:

- Topology Vectors (TVs) in a node can now coincide with the TKCs of its eligible neighbours. This reduces the memory occupation in nodes when pair-wise communications are adopted;
- in TAKS2, the transmission protocol uses only a single phase for the setup of the scheme, i.e., no prior exchange of the TKCs between nodes is needed. This is possible thanks to the $d$ information contained in each message;
- in TAKS2, authentication is performed by a standard authentication function (e.g., HMAC).

The main drawback in TAKS2 is the ephemeral TKC to be transmitted each time a new $SS$ is needed. Although this increases message size and thus energy consumption per transmission, in monitoring applications, the sample rate depends on the dynamics of the monitored system: if high transmission rates are needed, key size (hence message size) should be reduced without degrading security. In this context, as described in the previous section, TAKS2 can adopt ECC in order to reduce key size [7].

## *Formal apparatus of TAKS/TAKS2*

Building blocks of the proposed scheme are:

1. Hybrid-key cryptography
2. Topology-based network authentication

*Hybrid-key cryptography* Let nodes $n_i$ and $n_j$ be a pair of nodes. The following definitions are assumed:

a) Let $GF(p)$ be a prime-based Galois Field such that $p \gg N$ with $N$ equal to the number of nodes in the network and $p$ a large prime number such that $p-1$ has a suitable large prime divisor ($p$ should be a *safe* prime number). Otherwise, let $GF(p^n)$ be an extended Galois Field with $p$ prime (e.g. $p = 2$), $n$ an integer and $p^n \gg N$. Hereinafter simply $GF(\cdot)$;
b) Let $U$ a vector space of dimension $d$ (suppose $d = 3$). Let $KL \subseteq U$, $KT \subseteq U$ and $KV \subseteq U$;
c) Let $\overline{u} = (u_1, u_2, u_3, ..., u_d)$ be a vector over $GF(\cdot)$ and $k\overline{u} \in GF(\cdot) = (ku_1, ku_2, ..., ku_d)$ for $k = 1, 2, 3, ...p-1$;
d) Select $\overline{a_i}, \overline{a_j}, \overline{m} \in U$ such that $(\overline{a}_i \times \overline{a}_j) \neq 0$. $\overline{a_i}, \overline{a_j}, \overline{m}$ are *secret*. The set $A = \{a_i\}$ defines the *TAKS node IDs* *
e) Let $b \in GF(\cdot)$ be a multiplicative generator in $GF(\cdot)$. $b$ is secret;
f) Let $\overline{c} \in U$. $c$ is secret;
g) Parameters $A, b, \overline{c}, \overline{m}$ are called *TAK generators*;
h) Let $\overline{LKC} \in KL \subseteq U$ the Local Key Component. $LKC$ is secret;
i) Let $\overline{TKC} \in KT \subseteq U$ the Transmit Key Component. $TKC$ is public;

*These IDs are used only during the key components computation

j) Let $\overline{t} \in KV \subseteq U$ the Topology Vector. $\overline{t}$ is public;
k) Let $\overline{LKC}, \overline{TKC}, \overline{t}$ be result of public hardly invertible (i.e. *one-way*) functions of the TAK generators $(A, b, \overline{c}, \overline{m})$;
l) Let $U'$ be a vector space same dimension of $U$ which elements $\overline{u'} = (u'_1, u'_2, ..., u'_d)$ are in $\mathbb{Z}$;
m) Let $gf2z : U \to U'$ be a function which convert any $\overline{u} \in U$ in a correspondent $\overline{u'} \in U'$ such that $u = u'$;

n) Let: $\overline{a'_i} = gf2z(\overline{a_i}), \overline{a'_j} = gf2z(\overline{a_j}), \overline{m'} = gf2z(\overline{m})$ and $\overline{c'} = gf2z(\overline{c})$;

*Pair-wise TAKS2* In pair-wise communications, the scheme can be simplified using the following additional definitions:

o) Let $f(\overline{u'}) = kb^{\overline{m'} \cdot \overline{u'}}$ be a scalar function where $\overline{m'} \in U'$ satisfied (d) and $k \in GF(\cdot)$ is an arbitrary constant.
p) Let be $\overline{s}_i = \overline{m} f(\overline{a}_i)$ and $\overline{s}_j = \overline{m} f(\overline{a}_j)$.
q) Let $\overline{k}_{li} = \overline{LKC_i}$, $\overline{k}_{lj} = \overline{LKC_j}$, $\overline{k}_{ti} = \overline{TKC_i} = \overline{t}_i$, $\overline{k}_{tj} = \overline{TKC_j} = \overline{t}_j$ be defined as:

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

r) Expressions for $\overline{k}_l$ , $\overline{k}_t$ and $f(\overline{u'})$ are public according Kerckhoffs principle.

**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 TAKS node IDs in $A$ compliant to definitions (d) and be $f(\overline{u})$ a function defined as (o.) and $\alpha$ a random value in $GF(\cdot)$. If expressions for $\overline{k}_{li}$, $\overline{k}_{ti}$ and for $\overline{k}_{lj}$, $\overline{k}_{tj}$ are the same as (n) 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.** 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 $TAK_j$:

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

Using the vector algebra property $\overline{x} \cdot (\overline{y} \times \overline{z}) = \overline{y} \cdot (\overline{z} \times \overline{x})$, we have:

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

So, in TAKS2:

- the transmitter TAK is the scalar product between the LKC and the TV associated to the destination node;
- the receiver TAK is the scalar product between the LKC and the TKC.

Furthermore, fixed $\overline{m}$, $\overline{c}$, $b$ and be $\overline{a}_i$, $\overline{a}_j \in A$ the following properties hold:

1. $TAK \neq 0$ since $\overline{m'} \cdot (\overline{a}_i \times \overline{a}_j) \neq 0$ from (b) and $f(\overline{u}) \neq 0$ from **R1**.
2. The LKC components are different for different nodes since $\overline{k}_{li} \parallel \overline{a}_i$ 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. The TKC components are different for different nodes since $\overline{k}_{ti} \parallel \overline{m'} \times \overline{k}_{li}$ and $\overline{k}_{tj} \parallel \overline{m'} \times \overline{k}_{lj}$ and $\overline{k}_{li} \times \overline{k}_{lj} \neq 0$.

*Topology-based authentication* As in [8], TAKS/TAKS2 authentication is based on two main elements: a verification function $g(\cdot, \cdot)$ and a set of TVs, one for each eligible neighbours nodes of $n_i$.

With the definition of $TV(i) = \{TKC_{\sigma(i)}\} = \{\overline{k}_{t\sigma(i)}\}$ as the set of $\sigma(i)$ TV stored in node $n_i$, we can directly set $\overline{t}_j \equiv \overline{k}_{tj}$.

Given a user-selected keyed hash function [9] $MAC(\cdot)$, we define the authentication function as $g(p, v) = g(SS_j, SS_i) = MAC(SS_j) - MAC(SS_i)$. This latter definition is compliant with the **R4** and **R5** requirements define in the previous section.

**Theorem 2.** Network Topology Authentication. *In a node pair $n_i$ and $n_j$, if $MAC(SS_j)$ computed by receiver $n_j$ is equal to $MAC(SS_i)$ sent 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 [9] 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 $n_j$.

## Security Analysis

Security analysis can be performed along three proofs: entropy highness of the secret share, robustness (or reliability) of data transmissions for a single node (node level) and for a set of connected nodes (network level). The former deals with the goodness and effectiveness of the generated TAK as a *cryptographic key* for an high entropy encoding. The latter with the complexity of the reverse problem associated to the cryptosystem to derive the private key materials from publicly known parameters: the lower the complexity, the lower the reliability can be assured by the cryptosystem in data transmission: at a node level this means the evaluation of the computational complexity of the reverse problem. However in case of a compromised node (e.g. the node has been physically captured by the attacker), next problem would become how reliability in data transmission for other nodes could be reduced in case one or more nodes were compromised. Therefore robustness at network level can be straightforwardly defined as the maximum percentage

of compromised nodes that cannot reduce reliability in data transmissions for the other nodes in the network.

Let's start with TAK entropy evaluation. TAK entropy relies on the randomness of the secret TAKS primitives $\overline{m}$, $\overline{c}$, $\overline{a}$ and $b$. By definition these primitives are randomly chosen. Key components $\overline{k_l}$ and $\overline{k_t}$ are injective functions of secret TAKS primitives as well as TAK function defined as the scalar products, again an injective function, between vectors $\overline{k_l}_i$ and $\overline{k_t}_j$ or, equivalently, between vectors $\overline{k_l}_j$ and $\overline{k_t}_i$: therefore randomness is preserved from TAKS primitives to TAK. Randomness of $\overline{k_t}_i$ ($\overline{k_t}_j$) can be further enhanced by a random multiplicative factor to be locally applied to build the ephemeral $\overline{k_t}$ is being transmitted from node $i$ (node $j$) to node $j$ (node $i$). Therefore particular care has been used in designing and implementing the *Pseudo-Random Number Generator* algorithm in the nodes.

For what concerns TAKS robustness at single node and network levels, the following proofs can be provided.

**Scheme robustness at single node level:** in TAKS2 the reverse problem derives the secret TAKS primitives $\overline{m}$, $\overline{c}$, $\overline{a}$ and $b$ from the knowledge of $\overline{k_l}$, $\overline{k_t}$ and the (public) expression of $f(\cdot)$. Prudentially we include also $\overline{k_l}$ even if this is the private key component because both $\overline{k_l}$, $\overline{k_t}$ are preloaded in the node and therefore exposed to attacks ranging from power analysis or side-channelattacks to physical capture.

**Proof.** Each equation in (q) shows that the functional relationship between $\overline{k_l}_j$, $\overline{k_t}_i$, $\overline{m}$, $\overline{c}$, $\overline{a}$ and $b$ to be a discrete logarithm which is a problem of well-known complexity by cryptologists (also denoted as the Diffie-Hellman problem).

$$\overline{k_l} = \overline{a} b^{\overline{m} \cdot (\overline{a} + \overline{c})}$$
$$\overline{k_t} = (\overline{m} \times \overline{a}) b^{\overline{m} \cdot \overline{a}}$$

**Scheme robustness at network level** Given a network with N nodes, a cryptographic scheme is $T$-*secure* if an attacker should capture at least $T+1 < N$ nodes and therefore should compromise at least $T+1$ sets of their preloaded key materials to gain enough information to derive the other $N-(T+1)$ key materials preloaded in the other nodes of the network or, in other words, compromising only $T$ sets of preloaded key materials cannot reduce the reliability in data transmissions for the other $N-T$ nodes.

The maximum value for $T$ depends on the capability of network protocols to rearrange connection among the residual $N-T$ nodes without degradation of the overlying services. However usually is provided the proof for the case $T = N$ so that any choice for $T$ is covered. It can be proved that TAKS is $N$-*secure* (i.e., $T = N$).

**Proof.** Equations in (q) should be both solved for $\overline{a}$ , $\overline{m}$ , $\overline{c}$ and b to derive $\overline{k_l}$ , $\overline{k_t}$ and therefore all TAKs in the network. According to the definition of $T$-*Security*, the proof is iteratively developed: in case node $i$ has been captured, equations in (q) return $\approx p^4$ free solutions for ($\overline{a}$ , $\overline{m}$ , $\overline{c}$, $b$), hence $\approx p^4$ spurious solutions for TAK. A solution is defined spurious when part of it is computed by assigning arbitrary values to some unknowns due to lack of information on them (in this case $\approx p^4 - 1$ constraints are missing) and the corresponding TAK results incorrect and useless for the attacker. Therefore another node, say node $j = 2$, should be captured to compromise another set of key materials: equations in (q) now return $\approx p^4 - p$ free solutions for ($\overline{a}$ , $\overline{m}$ , $\overline{c}$, $b$), hence $\approx p^4 - p - 1$ spurious solutions for TAK. Note that the constraint $a_i \times a_j \neq 0$ indicates that $a_i$ and $a_j$ cannot be parallel which eliminates $\approx p$ solutions for ($\overline{a}$ , $\overline{m}$ , $\overline{c}$, $b$). Next step for the node $j = 3$, equations (q) would return $\approx p^4 - 2p$ free solutions for ($\overline{a}$ , $\overline{m}$ , $\overline{c}$, $b$), hence $\approx p^4 - 2p - 1$ spurious solutions for TAK. Note that the constraint $a_i \times a_j \neq 0$ shall be applied twice (for nodes $j$ with $j = 2$ and $j = 3$) and $\approx 2p$ solutions for ($\overline{a}$ , $\overline{m}$ , $\overline{c}$, $b$) shall be eliminated.

Iteratively if the node $j = n$ has been captured and its set of key materials compromised, equations in (q) return $\approx p^4 - np$ free solutions for ($\overline{a}$ , $\overline{m}$ , $\overline{c}$, $b$), therefore still $\approx p^4 - np - 1$ spurious solutions for TAK: as $N \ll p$ , there are still $p^4 - Np \approx p^4 - 1$ spurious solutions for TAK even for $n = N$.

**Table 3.** Computational complexity of TAKS2 functions.

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

## Computational and spatial complexity analysis

In this section, we describe the computational and spatial complexity of proposed scheme regardless of which symmetric encryption/decryption algorithm and MAC function are adopted. Table 3 shows the computational complexity of TAKS2. Also, assuming that:

- $n$ is the adopted key size (in bytes);
- $\#\sigma(i)$ is the cardinality of the set $\sigma(i)$;
- data is made of words of 1 or multiple bytes;
- the generation of a *random* word costs $O(1)$;
- the sum of two words costs $O(1)$;
- the multiplication of two words costs $O(1)$.

Then:

- to generate a $n$-bytes random byte-stream ($rand()$), we need to generate $\frac{1}{s}O(n)$ random words, so the cost is $O(n)$ ($s$ is the size of the word in bytes)
- to add two $n$-bytes number, we need to perform $O(n)$ additions of word, so a full addition costs $O(n)$;
- to multiply two $n$-bytes number, we need to perform $O(\frac{n(n+1)}{2})$ multiplications and $O(n)$ additions. So a multiplication costs $O(n^2)$;
- an *inner product* of two vectors of $n$-bytes components, consists of 3 $n$-bytes multiplications and 2 $n$-bytes additions, so an inner product costs $O(n^2)$;
- to compute TAK at the destination node we need to search for the right TKC/TV (e.g., linear search: $O(\#\sigma(i))$) and to compute the inner product with the LKC, for a total cost of $O(n^2 + \#\sigma(i))$.

In a point-to-point communication scenario, since sending a complete message using TAKS2 consists of executing the functions $rand()$, $get\_tak()$ and $multiply()$, the total computational complexity of TAKS2 encryption is

$O(n^2 + \#\sigma(i)) \simeq O(n^2)$. The decryption step performed by the destination node has a complexity equal to the complexity of an $inner\_product()$, so $O(n^2)$.

The spatial complexity is analyzed as follows. Assuming:

- each Key Component (LKC, TKC) requires $k$ bytes to be stored in the node memory;
- the $\alpha$ random value, the key reconstruction information $KRI$ and the $SS$ require $k$ bytes each;
- each node has to store its LKC and the TKCs of each neighbour node inside the PNT.

We can state that:

- $\alpha$, $KRI$, SS and $\tau$ require a constant memory size ($O(1)$);
- each node has to reserve an additional space of $O(k + n * k) = O(n * k)$ for storing the PNT, where $n$ is the maximum number of neighbour nodes (i.e., the maximum *degree* that can be found in the WSN topology graph).

## Design and Implementation Issues

The topic of this section is the design and implementation of TAKS2 for a real-world star topology WSN. Firstly, a basic background on 802.15.4 is provided, followed by specific considerations for TAKS2; then, implementation issues are analyzed for software architecture and details about two different available MAC layers are explained.

### *Introduction to 802.15.4*

The IEEE 802.15.4 [27] is the reference standard on Low Rate Wireless Personal Area Networks (abbr. LR-WPAN). It describes the first two layers of the ISO/OSI networking protocol stack (the *Physical Layer* and the *MAC Layer*). The IEEE 802.15.4 defines two types of nodes: the *Full Function Devices* (FFD) and the *Reduced Function Devices* (RFD). While the RFD are devices capable to perform only basic computation and data retrieval, the FFD are more complex and computational powerful devices, capable also to act as *WPAN Coordinator* of the network. The *WPAN Coordinator* (i.e., *Coordinator*) is the device responsible of creating, maintaining and synchronizing the network.

The standard defines some basic network *topologies* (e.g. basic *peer-to-peer*, *star topology*, *cluster tree*) and two WPAN types. The nodes in a *Beacon-Enabled* WPAN have an isochronous channel access obtained through the use of *beacon* frames emitted by the Coordinator. Instead, in a *Beaconless* WPAN the nodes communicates without such frames. In both cases, the *CSMA-CA* channel access mechanism is used: in the case of beacon-enabled PANs, the slotted CSMA-CA is used, while the unslotted CSMA-CA (ALOHA) is used in the beaconless PANs.

In the case of beacon-enabled PANs, the standard defines a *Superframe* structure (Figure 2). This structure can be further divided into the *active* and the *inactive* portion. Inside the active portion, a set of optional *Contention-Free* slots can be defined to reserve communication slots to a chosen set of devices (up to 7).

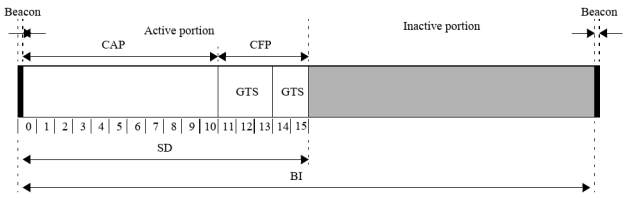


**Figure 2.** The IEEE 802.15.4 Superframe structure [27]

Each of the two provided layers offers a set of *services* via a corresponding set of interfaces. The services offered by the PHY layer to the MAC layer are the *PHY Data service* (`PD`) and the *PHY Management Service* (`PLME`). The services provided by the MAC layer to the higher layers are the *MAC Data Service* (`MCPS`) and the *MAC Management Service* (`MLME`).

While the IEEE 802.15.4 PHY layer is commonly implemented in radio chips, the MAC layer specification has not a wide selection of available implementations. This is due the fact that applications often use simpler techniques for accessing the communication channel instead of implementing a 802.15.4 compliant MAC layer. Despite this aspect, in literature exist some valid implementations of the 802.15.4 MAC layer. In this paper we consider two well-known implementations: the *TKN154* [29] and the *Atmel Stack MAC layer* [30].

### *TAKS2 design considerations*

In order to provide an efficient TAKS2 architecture design, the features of the IEEE 802.15.4 standard have been taken into consideration in the design phase.

First of all, in order to provide a seamless and efficient integration with the IEEE 802.15.4 standard, TAKS2 is designed to operate directly at the MAC layer although it is possible, in general, to implement TAKS also in upper layers. TAKS2 cryptographic primitives are made available to the MAC layer services, e.g., during the device association, to achieve a better control over incoming and outgoing transmissions and smaller latency during encryption, decryption and authentication.

As a consequence, a device inside a TAKS2-enabled WSN has to pass two kinds of *association* before being able to transmit and receive data from the coordinator. The first is the IEEE 802.15.4 MAC association step (via the MLME-ASSOCIATE interface defined in the standard), while, the second is the *TAKS2 association*, a primitive that enables a device-coordinator link to be validated through the TAKS2 authentication function defined in the previous sections.

TAKS2 uses a set of *Finite State Machines* (FSMs) (Figure 3) to control the behavior of cryptographic primitives. In particular, each device node has a tri-state machine with the following states:

- **UNKNOWN** - the device is not currently associated to the coordinator
- **ASSOCIATED** - the device is associated (via the IEEE 802.15.4 association mechanism) to the coordinator
- **READY** - the associated device is authenticated also via the *TAKS2 association*. In this state, the device can exchange messages with the coordinator.

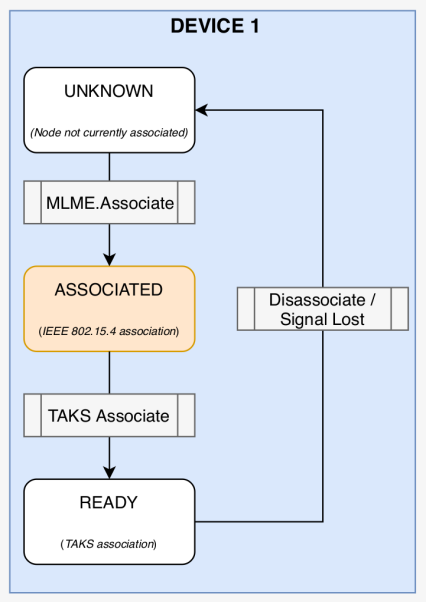


**Figure 3.** FSM used by TAKS2 for the device nodes

The coordinator keeps and updates a copy of the state of each device FSM, so that is also possible for it to monitor the overall WSN state. TAKS2 uses a set of additional data structures (Figure 4) to support the IEEE 802.15.4 standard:

- The **Address Pool** represents the set of available device addresses. Each entry in the data structure contains the short and extended 802.15.4 address and an availability flag.
- The **Neighbor Table** stores a set of *Neighbor Control Block*s (NCB), one for each device. An NCB contains the address information of the device (short and extended address), its FSM state and the private cryptographic information (key reconstruction information, SS and TKC/TVs).

## SW Architecture and Implementation Issues

In this section we describe two implementations of TAKS2 based on the design described in the previous section. One adopts the TKN154 MAC layer and the other is based on the Atmel proprietary stack. The main goal is to demonstrate the feasibility of using TAKS2 on top of well-known IEEE 802.15.4-compliant network stacks. In the following sections, the two implementations are discussed and the feasibility of introducing TAKS is analyzed. Then, both of them are verified and compared to highlight the advantages and disadvantages of each one, with particular attention to their *performance* results.

*Software Architecture* The software architecture adopted in both implementation is shown in Figure 5. The architecture uses the *Bridge* design pattern in order to provide a non-intrusive, de-coupled and flexible approach to connect the target MAC layer with the TAKS2 components. In Figure 5, the IEEE 802.15.4 MAC layer (bottom box) offers two of the available services (`MCPS-Data` and `MLME-ASSOCIATE`) to the *Bridge*, which calls the TAKS2 key generator and the inner symmetric block cipher to provide to the upper layer the TAKS2 primitives. For sake of example, in the figure, the MAC layer shown is based on the TKN154 while the block cipher is based on the *Advanced Encryption Standard* (AES) cipher in CBC mode with a key length of 192 bits. The *Bridge* layer takes care of: monitoring MAC layers requests; building proper responses; retrieving the TAKS2 key components and combining them according the scheme to generate the *Shared Secret*, which is then used as symmetric key in the AES encryption/decryption. Such an architecture allows TAKS2 implementations to adapt and evolve, supporting new features and new symmetric ciphers without the need of a code refactoring process. Also, since the *Bridge* layer can optionally be disabled, it offers also a compile-time *On-Demand* security option, so that a WSN network operator can choose which node has to use TAKS2 (increasing the communication security) or to use direct MAC layer primitives (increasing the performances) seamlessly.

*TKN154* The *TKN154* is an open-source implementation of the 802.15.4 MAC layer from the *Telecommunication Network Group* of the *Technical University of Berlin*. The TKN154 is based on *TinyOS 2.x* operating system, it is written in the *nesC* language and it is compatible with node platforms which use the CC2420 radio chip (e.g. *Telosb*, *T-Mote*, *MicaZ*). The TKN154 offers four major features:

1. the TKN154 source code is released under an open-source license;
2. the code is organized to be as independent as possible from the underlying hardware platform. Hardware-dependent code is separated through interfaces from the rest of the code;
3. most of the features of the 802.15.4 standard are kept in separated components, making possible to enable or disable them selectively;
4. the whole TKN154 architecture is organized in various abstraction levels, allowing it to be highly customizable.

One drawback of the TKN154 is the *pre-requisites* needed by a platform to be fully compatible with it. The most evident of these prerequisites is a proper and stable hardware timer, which should be able to output a 62.5 KHz frequency clock with a maximum error of 40ppm.

*Atmel MAC* The TKN154 project is not the only 802.15.4 MAC implementation available: Atmel provides a proprietary *network stack* [30] for its AVR platforms which complies on the 802.15.4 standard. The stack is organized into sub-layers. Among those, the *Platform Abstraction Layer* (PAL), the *Transceiver Abstraction Layer* (TAL) and the *MAC Core Layer* (MCL) constitute the part of the stack that can be seen as the MAC layer. The first provides hardware-dependent code needed to interface to available peripherals, the TAL takes care of frame handling, power management, CSMA handling and general state transitions while, the MCL implements the basic 802.15.4 MAC standard primitives, such as the *MAC Management Service* (MLME).

*Random Number Generation* The $\alpha$ value in the scheme has to be randomly generated to ensure it is unpredictable by attackers. This aspect is critical since a vulnerability in

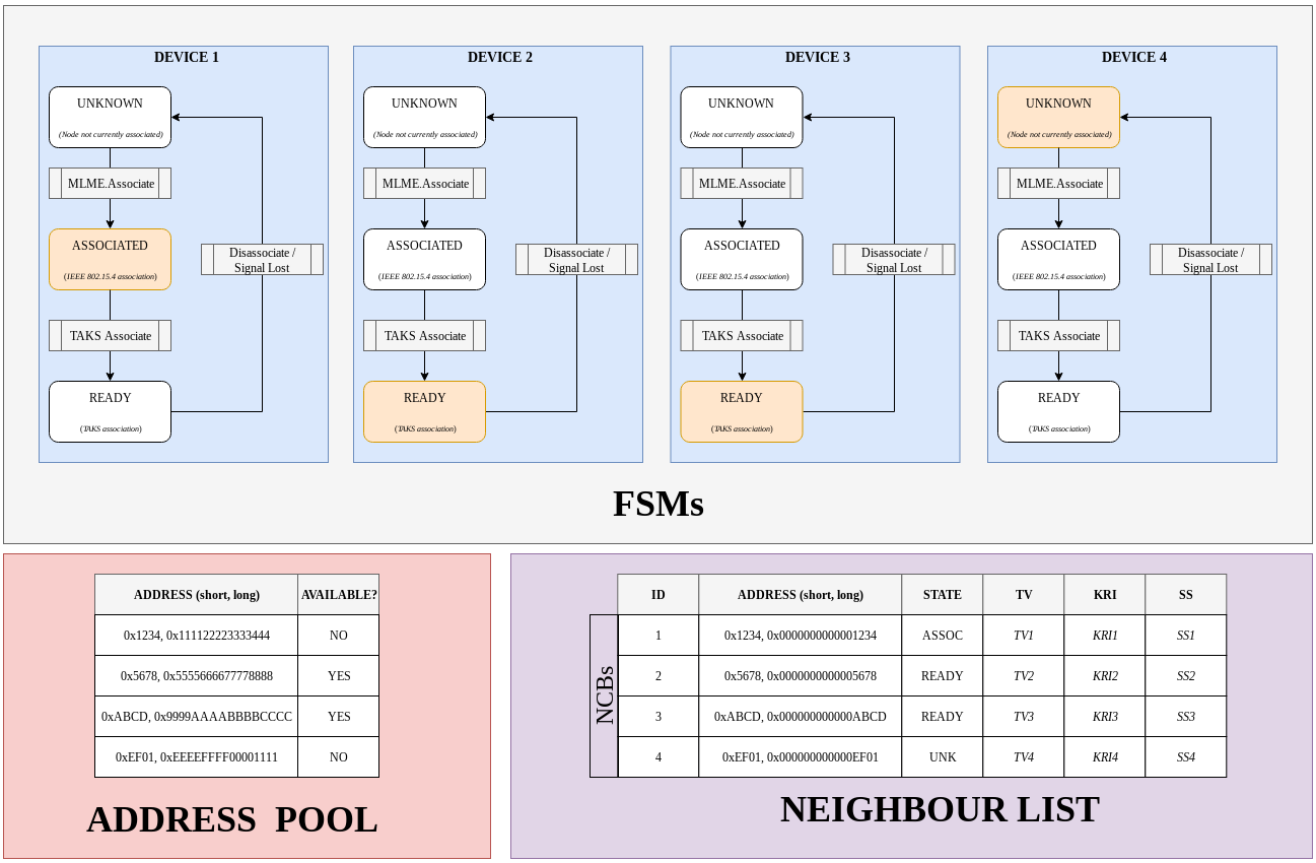


**Figure 4.** TAKS2 data structures

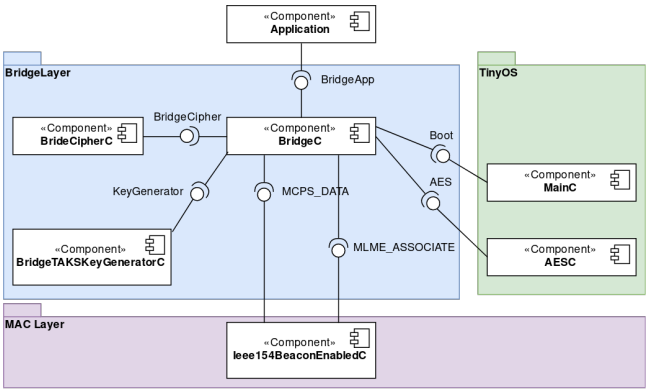


**Figure 5.** TAKS2 implementations common architecture

the random number generation can affect the security of the overall scheme. In particular, $\alpha$ has to be generated by a Pseudo-Random Number Generator (PRNG) which has some specific properties that make it suitable for cryptography (CSPRNG). A PRNG is defined by a deterministic algorithm which is able to generate, given an input value called *seed*, a pseudo-random numbers sequence. The output sequence of a PRNG, *approximately*, has the same statistical properties of a sequence generated by a truly random process. In the cryptography context, it is fundamental that the PRNG generates a pseudo-random numbers sequence which is unfeasible to be entirely reconstructed starting from a sub-sequence of it: an attacker could listen to the sequence, retrieve the seed and hence, learn the future pseudo-random values. In TAKS2, the $\alpha$ value in the scheme is generated by a PRNG based on *Mersenne Twister* algorithm [37]. In order to increase randomness of the value, we have used the *sensing functionality* of WSN nodes (i.e., the temperature and the humidity sensors) combined with LQI and RSSI values coming from the radio transceiver. Given the unfeasibility to learn about the values of the physical phenomena in the area around the WSN nodes, we have used such values to create the *seed* for our implementation of the Mersenne Twister algorithm.

## Verification and Performance Evaluation

In order to check the correctness of the implementations of the scheme and their performance, in this section we describe the verification phase of TAKS2. A set of scenarios, each one using a different operating conditions for devices and coordinator, has been selected to test TAKS2 components. For each scenario, we have verified the correctness and the effectiveness of the TAKS2 implementations when deployed in a star topology[†].

### *Basic Network Configuration*

The basic network configuration used in the verification phase (Figure 6) consists of a *star-topology* WSN formed by one cluster head node (*coordinator*) connected to nine different cluster member nodes (*device*).

[†]The verification of the basic point-to-point configuration can be found in [36]

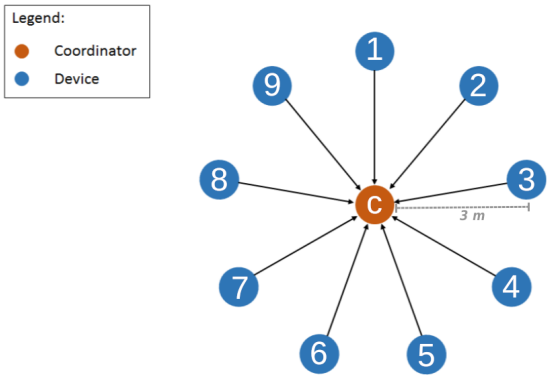


**Figure 6.** Performance Evaluation Network Topology.

**Figure 7.** CM5000 WSN Node

**Figure 8.** Atmel REB212BSMA: Node

This is a common topology that is used in data collection environments, where the cluster members are meant to retrieve data while the cluster head collects, elaborates and transmits them.

The sensor nodes used in the experiments are the following:

- for the TKN154-based implementation, we have adopted the *Advanticsys CM5000* [31] (Figure 7), a *clone* of the more famous *TelosB* [34] node. This node embeds a MSP430 micro-controller unit, a CC2420 radio chip (IEEE 802.15.4 compliant), 48 KiB of Flash memory and 10 KiB of RAM;
- for the Atmel MAC implementation, we have adopted the nodes included inside the *Atmel REB212BSMA-EK* kit [32] (Figure 8). They are based on the Atmel ATMega1281 micro-controller unit with 128 KiB of Flash, 8 KiB of RAM memory and the AT86RF212B (IEEE 802.15.4 compliant) radio transceiver.

*Testbed application* During the experiments, the coordinator and the devices have been programmed with a *testbed application*. This testbed performs a typical monitoring application using *directly* the interfaces provided by the MAC layer. In this way, we can verify the correctness of TAKS2 primitives and the communication between TAKS2 and the MAC layer in a *realistic environment*. In such a testbed the nodes have the following roles:

- each device periodically samples its installed sensors (i.e., temperature and humidity), builds a message with the retrieved measures and sends it to the coordinator;
- the coordinator receives messages, parses their content, aggregate the measurements and sends the result to the PC via the UART port.

This testbed is deployed in two flavors, the *un-secure* version (i.e., no TAKS2) and the *secure* version (i.e., using TAKS2). The two versions differ on the components involved in the communication: the un-secure testbed communicates by using no additional layers, while the secure version uses TAKS2 via the *Bridge* layer discussed in Section 4. The two versions of the testbed have been deployed and tested on the CM5000 WSN nodes (using the TKN154 MAC) and on the REB212BSMA kit nodes (using the Atmel 802.15.4 MAC). The discussion of the obtained results is reported in the following paragraphs.

## Scenarios description

In order to verify the implementations and to evaluate performances, we have used both the *un-secure* and *secure* versions of the testbed application against five different scenarios, each one with a different set of *network parameter*[‡] values. The network parameters we have considered are shown in Tab. 4.

The selected length (in bits) for the elements in $GF(p)$ and the key components (LKC, TKC) has been set to *192*, which it is a good compromise between memory size and security level.

*Verification Results* The verification of both TAKS2 implementations with the two testbed versions across the different MAC parameters sets has been successful: by analyzing the outputs of the communications between the coordinator towards the PC, it is possible to reasonably state that TAKS2 works as intended: the $SS$ reconstruction (on the receiver side) is correct, the authentication code is verified and the encrypted messages are successfully decrypted (Figure 9 shows a screen-capture of the TKN154 testbed). This result highlights the feasibility of adopting TAKS2 in real-world WSN-based monitoring applications in order to secure the communication among the sensor nodes.

## Performance evaluation

Following the initial verification, a performance evaluation has been conducted in order to analyze the *overhead* related to the usage of the scheme. So, in this section, we first introduce the *metrics* that have been defined to evaluate the performance, then we report some of the obtained results, mainly focusing on the scheme impact on performances.

*Metrics* In order to evaluate the performance of TAKS2 scheme implementations and the impact on the MAC layer performance, we have defined two set of *metrics*. The first set is composed of metrics related to the overall MAC layer performance:

- *MAC Layer Association (MAC Assoc)*. This is the time needed by a device node to successfully perform the IEEE 802.15.4 association process (`MLME-ASSOCIATE`) with the coordinator.
- *Transmission time (TX time)*. This is the delay between the MAC data request (`MCPS-DATA.request`) and actual transmission of the first frame of the message.

[‡]More information about these parameters can be found in [27]

**Table 4.** MAC parameters sets

| Scenarios: | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| *Beacon Interval* | 61440 | 30720 | 122880 | 61440 | 30720 |
| *Scan Duration* | 61440 | 30720 | 122880 | 61440 | 30720 |
| *Scan Duration (Coord.)* | 123840 | 123840 | 123840 | 62400 | 31680 |
| *Scan Duration Short* | 62400 | 31680 | 123840 | 62400 | 31680 |
| *Scan Duration Long* | 123840 | 62400 | 246720 | 123840 | 62400 |

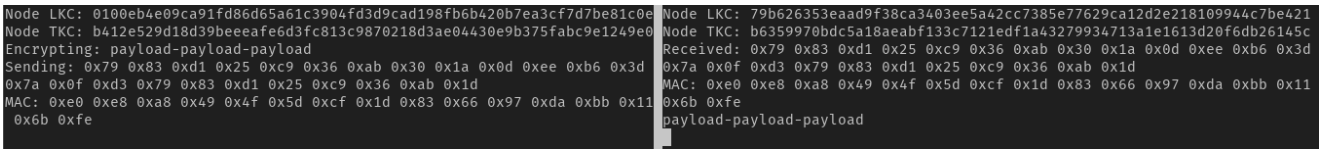


**Figure 9.** TAKS2: data from a sender node (left) and a receiver node (right) sent to PC via UART

- *Reception time (RX Time)*. This is the delay between the MAC layer data arrival (`MCPS-DATA.indication`) and when such data is passed to the higher layers.
- *Frame Error Rate*. This is the number of unsuccessfully received frames by the MAC layer with respect to the total transmitted frames.

The second set of metrics has been used to evaluate the performances of TAKS primitives (e.g., encryption, decryption, key generation, etc.):

- *TAKS2 key computation time (TX key gen)*. This is the time required by TAKS2 to successfully generate a *Shared Secret* from the key components.
- *TAKS2 key retrieval time (RX key gen)*. This is the time required by TAKS2 to successfully recreate the proper *Shared Secret* from the $KRI$ and the receiver's key components.
- *TAKS2 Association Delay (TAKS2 Assoc)*. This is the delay introduced by the *TAKS2 Association*, which takes place after the IEEE 802.15.4 default MAC association process (`MLME-ASSOCIATE`).

## Results

In order to correctly evaluate the MAC layer and TAKS2 performance, we have conducted tests considering all the parameters set reported in Table 4. The tests have been replicated for the un-secure and TAKS2-enabled versions of the testbed for both the MAC layer implementations. The results reported in this section are the average values of the metrics described in Section  performed across the different parameter sets and each device node. The results for the TKN154-based implementation are shown in Table 5 for *un-secure* testbed application and in Table 6 for the *secure* version, while, in Table 7 and 8 we show the results related to the Atmel Stack.The total memory overhead of the secure version for storing the LCD in our test scenario (Figure  6) has been *˜400* bytes for the cluster-head node and *˜200* bytes for the other nodes. The size of additional code required to implement the $TAK(\cdot)$ function has been *˜400* bytes for the TKN154 testbed and *˜300* bytes for the Atmel testbed. Additional memory is required by the adopted symmetric cipher (i.e., AES) implementation. In our case, for the TKN154 testbed application the memory required by the AES implementation (as a TinyOS component) has been *˜1300* bytes, while, for the Atmel-based testbed, the memory required has been *˜800* bytes.

**Table 5.** Un-secure stack (TKN154)

| | Min (ms) | Max (ms) | Avg |
|---|---|---|---|
| *Tx time* | 9.86 | 51.55 | 11.44 |
| *Rx time* | 6.34 | 6.75 | 6.34 |
| *MAC Assoc* | 16.59 | 48.35 | 23.89 |
| *Frame Error Rate (±0.05%)* | | 1% | |
| *Tx key gen* | - | - | - |
| *Rx key gen* | - | - | - |
| *TAKS2 Association* | - | - | - |

**Table 6.** Secure stack (TKN154)

| | Min (ms) | Max (ms) | Avg |
|---|---|---|---|
| *Tx time* | 35.84 | 85.41 | 39.34 |
| *Rx time* | 26.50 | 29.12 | 27.34 |
| *MAC Assoc* | 16.62 | 48.40 | 23.92 |
| *Frame Error Rate (±0.05%)* | | 1.8% | |
| *Tx key gen* | 13.31 | 13.70 | 13.44 |
| *Rx key gen* | 5.12 | 5.31 | 5.18 |
| *TAKS2 Assoc* | 11.26 | 32.42 | 13.26 |

**Table 7.** Un-secure stack (Atmel)

| | Min (ms) | Max (ms) | Avg |
|---|---|---|---|
| *Tx time* | 21.48 | 187.01 | 28.49 |
| *Rx time* | 0.62 | 0.67 | 0.63 |
| *MAC Assoc* | 16.69 | 48.68 | 27.34 |
| *Frame Error Rate (±0.05%)* | | 0% | |
| *Tx key gen* | - | - | - |
| *Rx key gen* | - | - | - |
| *TAKS2 Assoc* | - | - | - |

**Table 8.** Secure stack (Atmel)

| | Min (ms) | Max (ms) | Avg |
|---|---|---|---|
| *Tx time* | 61.62 | 203.82 | 71.67 |
| *Rx time* | 2.65 | 2.84 | 2.73 |
| *MAC Assoc* | 17.35 | 49.90 | 28.79 |
| *Frame Error Rate (±0.05%)* | | 0.15% | |
| *Tx key gen* | 35.23 | 177.21 | 42.31 |
| *Rx key gen* | 6.82 | 6.88 | 6.84 |
| *TAKS2 Assoc* | 17.35 | 49.9 | 28.79 |

**Result Analysis** — The results show that in the TKN154 secure version of the testbed, the performance loss introduced in Tx/Rx operations is around $20ms$ (from $+150\%$ to $+330\%$). In the same tables, it is possible to observe that TAKS2 does not interfere with the standard 802.15.4 MAC association. The frame error rate is almost the same for the un-secure and secure version of the testbed, although the longer transmission times reported in the secure version slightly affected the rate ($+0.8\%$). From the values, it is noticeable that the transmission time (*TX time*) in both the secure and un-secure results shows a higher deviation. This is due to different factors, although the higher contribution is related to the channel access protocol (CSMA). Lastly, in the

secure version, the *Tx key gen* and the *Rx key gen* show an average $8ms$ difference. This difference is expected, since the key computation step during transmission is different and requires more operations (e.g. the random number computation, the $SS$ retrieval, etc.). The Atmel MAC results show a relevant performance loss ($+100\%$) in transmission time respect to the TKN154 MAC in both secure and unsecure versions of the testbed. However, the reception time and the frame error rate are considerably better in the Atmel MAC. This shows that, probably, the Atmel MAC implementation trades longer transmission times for a better success rate. Moreover, as in the TKN154 version, the Atmel MAC secure version of the testbed shows a relevant difference between the *Tx key gen* and the *Rx key gen*. Finally, Atmel *Tx key gen* computation time is more than three times the TKN154 one. The reason of this difference is partly due to the different computations required during transmission, but also due to the different code style of the key generation primitives (i.e., the TKN154 uses TinyOS 2.x as lower-level code, while the Atmel MAC uses its own low-level code substrate).

Considering a typical monitoring application deployed on a WSN e.g. a star-topology with the local PAN coordinator acting also as sink-node a the centre, the performance overhead could be neglected if the transmission frequency is less than the maximum time required to sample, encrypt, associate and transmitting the measurements. As the results show, baseline TAKS2 implementation on common WSN nodes is effective in the case of slow-monitoring application. In order to make TAKS2 effective also for higher transmission frequency, the following solutions could be adopted while preserving the security level:

1. Encrypting and sending multiple samples per transmission;
2. Providing a platform-specific implementation of TAKS2 to gain advantages from the platform-specific features (e.g. hardware multiplicators);

On the memory impact, both implementations use less than *2* Kilobytes of additional memory to store data and code required by the scheme. Considering state-of-the-art cryptographic schemes and their implementation on WSN platforms (e.g. AES, RSA, ECC-ECIES), we can state that, generally, TAKS requires more memory than pure symmetric schemes (since TAKS itself requires a symmetric encryption scheme), but less than any public-key schemes. In fact, considering public-key schemes such as the ECC-based encryption scheme ECIES [22] and using the least possible memory, each node is still required to store, in addition to its private and public keys, at minimum, the curve domain parameters for each supported curve and the code required to implements elliptic curve basic point operations (in the order of tens of Kilobytes).

## Conclusions and future works

This paper has presented the evolution of a hybrid cryptography scheme, based on the generation of topology authenticated keys, specifically designed for WSN platforms. The scheme has been demonstrated formally by proving that the scheme is N-secure, which guarantees that attackers should retrieve N node keys before being able to compromise the security of the WSN. In order to validate the scheme, we have developed two implementation of TAKS2, based on two state-of-art IEEE 802.15.4 MAC layers. A *testbed* application has been developed to evaluate their performances. The validation and performance evaluation results have shown that the scheme provides security with a performance loss which is negligible in common environmental monitoring applications i.e. when slow-changing physical phenomena is monitored (e.g. temperature, light intensity, humidity, etc.). While TAKS2 is, by design, a lightweight scheme, yet the performance overhead introduced on the WSN nodes (which are resource-constrained by themselves) make TAKS2 less effective in those situations when an high frame throughput (along security) is required (e.g. audio/video streaming, high data-rate data exchanges etc.).
The work described in this paper is part of a wider research project that aims to create a secure framework for WSN to provide a encryption/decryption scheme associated with an *Intrusion Detection System* (IDS) [35]. In the future, we plan to implement the schema also for cluster-tree WSNs and to adapt it to be compliant to the KMP (Key Management Protocol) as defined in the recent IEEE 802.15.9 standard [28]. The final goal is to integrate all such features in a MW for WSN (e.g., Agilla2 [38]) able to provide efficient and effective security services.

## Acknowledgements

This work has been partially supported by the ECSEL RIA 2015 SAFECOP project.

## References

[1] Dinker, A.G., Sharma, V. Polynomial and matrix based key management security scheme in wireless sensor networks (2019) 22 (8), pp. 1563-1575. DOI: 10.1080/09720529.2019.1695904

[2] Camtepe, S. A., Yener, B.: Key distribution mechanisms for wireless sensor networks: a survey. Techical Report TR-05-07, Troy, 2005.

[3] Ahlawat, P., Dave, M. A resilient and seamless rekeying scheme based on random key distribution for WSN (2017) 2017 International Conference on Computer, Communications and Electronics, COMPTELIX 2017, art. no. 8003987, pp. 321-327. DOI: 10.1109/COMPTELIX.2017.8003987

[4] Lara-Nino, C.A., Diaz-Perez, A., Morales-Sandoval, M. Elliptic Curve Lightweight Cryptography: A Survey (2018) IEEE Access, 6, art. no. 8536394, pp. 72514-72550. DOI: 10.1109/ACCESS.2018.2881444

[5] Mukherjee, Nandini, Sarmistha Neogy, and Sarbani Roy.Building wireless sensor networks: theoretical and practical perspectives. CRC Press, 2017.

[6] Liao, X., Xu, S., Wang, S., Zhou, K. A method of pairwise key distribution and management in distributed wireless sensor networks (2007) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 4864 LNCS, pp. 834-844.

[7] Pugliese M., Pomante L., Santucci F.: Secure Platform over Wireless Sensor Networks. INTECH Publishers. 2012. ISBN 978-953-51-0218-2.

[8] Pugliese, M., Santucci, F.: Pair-wise Network Topology Authenticated Hybrid Cryptographic Keys for Wireless Sensor Networks using Vector Algebra. 4th IEEE International Workshop on Wireless Sensor Networks Security (WSNS2008), Atlanta, 2008.

[9] Rogaway, P., Shrimpton, T.: Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision-Resistance. FSE 2004, LNCS 3017, 371-388.

[10] Y. Liu, X. Wu and X. Chen, "A scheme for key distribution in wireless sensor network based on Hierarchical Identity-Based Encryption," 2015 IEEE 12th International Conference on Networking, Sensing and Control, Taipei, 2015, pp. 539-543. doi: 10.1109/ICNSC.2015.7116094

[11] K. Hamsha and G. S. Nagaraja, "Analysis of security mechanism using threshold cryptography for hierarchical wireless sensor networks," 2017 International Conference on Communication and Signal Processing (ICCSP), Chennai, 2017, pp. 1938-1941. doi: 10.1109/ICCSP.2017.8286737

[12] R. Kuchipudi, A. A. M. Qyser and V. V. S. S. S. Balaram, "A dynamic key distribution in wireless sensor networks with reduced communication overhead," 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), Chennai, 2016, pp. 3651-3654. doi: 10.1109/ICEEOT.2016.7755389

[13] H. Arjmandi and F. Lahouti, "A key pre-distribution scheme based on multiple block codes for wireless sensor networks," 7'th International Symposium on Telecommunications (IST'2014), Tehran, 2014, pp. 857-860. doi: 10.1109/ISTEL.2014.7000823

[14] R. J. Kavitha and B. E. Caroline, "Hybrid cryptographic technique for heterogeneous wireless sensor networks," 2015 International Conference on Communications and Signal Processing (ICCSP), Melmaruvathur, 2015, pp. 1016-1020. doi: 10.1109/ICCSP.2015.7322653

[15] Manjunath CR, S. Anand and G. Nagaraja, "An hybrid secure scheme for secure transmission in grid based Wireless Sensor Network," 2015 IEEE International Advance Computing Conference (IACC), Banglore, 2015, pp. 472-475. doi: 10.1109/IADCC.2015.7154753

[16] D. Kim, J. Yun and S. Kim, "Hybrid public key authentication for wireless sensor networks," 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST), Cambridge, 2017, pp. 142-143. doi: 10.23919/ICITST.2017.8356364

[17] Shim, K.-A. A survey of public-key cryptographic primitives in wireless sensor networks (2016) IEEE Communications Surveys and Tutorials, 18 (1), art. no. 7172449, pp. 577-601. DOI: 10.1109/COMST.2015.2459691

[18] R. K. Kodali, "Implementation of ECC with hidden Generator point in Wireless Sensor Networks," 2014 Sixth International Conference on Communication Systems and Networks (COMSNETS), Bangalore, 2014, pp. 1-4. doi: 10.1109/COMSNETS.2014.6734924

[19] Rijndael proposal (AES) https://csrc.nist.gov/csrc/media/projects/cryptographic-standards-and-guidelines/documents/aes-development/rijndael-ammended.pdf

[20] R. L. Rivest, A. Shamir, and L. Adleman. 1978. A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM 21, 2 (February 1978), 120-126. DOI=http://dx.doi.org/10.1145/359340.359342

[21] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in IEEE Transactions on Information Theory, vol. 31, no. 4, pp. 469-472, July 1985. doi: 10.1109/TIT.1985.1057074 keywords: Cryptography;Logarithmic arithmetic, URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1057074&isnumber=22749

[22] Agrawal, Lalita, and Namita Tiwari. "A Review on IoT Security Architecture: Attacks, Protocols, Trust Management Issues, and Elliptic Curve Cryptography."Social Networking and Computational Intelligence. Springer, Singapore, 2020. 457-465.

[23] TinyOS Home Page, http://www.tinyos.net

[24] ContikiOS Home Page, http://www.contiki-os.org/

[25] RIOT-OS Home Page, https://www.riot-os.org/

[26] Gudivada, R.B., Hansdah, R.C. Energy efficient secure communication in wireless sensor networks (2018) Proceedings - International Conference on Advanced Information Networking and Applications, AINA, 2018-May, art. no. 8432257, pp. 311-319. DOI: 10.1109/AINA.2018.00055

[27] IEEE Standard for Low-Rate Wireless Networks, in IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011) , vol., no., pp.1-709, April 22 2016 doi: 10.1109/IEEESTD.2016.7460875, URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7460875x&isnumber=7460874

[28] IEEE Recommended Practice for Transport of Key Management Protocol (KMP) Datagrams," in IEEE Std 802.15.9-2016 , vol., no., pp.1-74, Aug. 17 2016 doi: 10.1109/IEEESTD.2016.7544442, URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7544442&isnumber=7544441

[29] J. Hauer, "TKN15.4: An IEEE 802.15.4 MAC Implementation for TinyOS 2", TKN Technical Report TKN-08-003, Berlin, March 2009, URL: http://www.tkn.tu-berlin.de/fileadmin/fg112/Papers/TKN154.pdf

[30] Atmel (now Microchip) 802.15.4 MAC website, http://www.microchip.com/design-centers/wireless-connectivity/embedded-wireless/802-15-4/software/ieee-802-15-4-mac

[31] MTM-CM5000-MSP product website, https://www.advanticsys.com/shop/mtmcm5000msp-p-14.html

[32] Atmel REB212BSMA Hardware User Manual, https://www.mouser.com/datasheet/2/268/Atmel-42097-WIRELESS-AT02876-REB212BSMA-Hardware-U-1368556.pdf

[33] Atmel IEEE 802.15.4 libraries https://www.microchip.com/design-centers/wireless-connectivity/embedded-wireless/802-15-4/software/ieee-802-15-4-mac

[34] TelosB product datasheet, http://www.memsic.com/userfiles/files/Datasheets/WSN/telosb_datasheet.pdf

[35] L. Pomante, W. Tiberti, M. Pugliese, M. Santic, L. Di Giuseppe, L. Bozzi, F. Santucci. "TinyWIDS: a WPM-based Intrusion Detection System for TinyOS2.x/802.15.4 Wireless Sensor Networks". Fifth Workshop on Cryptography and Security in Computing Systems (CS2), Manchester (UK), January 2018

[36] L. Pomante, M. Pugliese, S. Marchesani, F. Santucci. "Definition and Development of a Topology-based Cryptographic Scheme for Wireless Sensor Networks". Sensor Systems and Software - Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, 2013.

[37] Saito, Mutsuo, and Makoto Matsumoto. "SIMD-oriented fast Mersenne Twister: a 128-bit pseudorandom number generator." Monte Carlo and Quasi-Monte Carlo Methods 2006. Springer, Berlin, Heidelberg, 2008. 607-622.

[38] Agilla2 Github project page: https://github.com/luigi-pomante/Agilla2

[39] Tongxu Yue, Chuang Wang and Zhi-xiang Zhu, Hybrid Encryption Algorithm Based on Wireless Sensor Networks, Proceedings of 2019 IEEE International Conference on Mechatronics and Automation August 4 - 7, Tianjin, China

[40] Zhang Ying, Ji Pengfei, An Efficient and Hybrid Key Management for Heterogeneous Wireless Sensor Networks, 978-1-4799-3708-0/14/$31.00 2014 IEEE

[41] Attiya Akram, Ayesha Naureen, Rabia Riaz, Ki Hyung Kim, Hafiz Farooq, Evaluation of Hybrid Security System with Cluster Based key Management for Wireless Sensor Networks, International Conference on Intelligent and Advanced Systems 2007

[42] Manjunath CR, Sindhu Anand, GS Nagaraja, An Hybrid Secure Scheme for Secure Transmission in Grid Based Wireless Sensor Network

[43] Tahir, Ruhma & Javed, Muhammad & Ahmad, Attiq & Raja, Iqbal. (2008). SCUR: Secure Communications in Wireless Sensor Networks using Rabbit. Lecture Notes in Engineering and Computer Science. 2017.