Multichannel Protocols for User-Friendly and Scalable Initialization of Sensor Networks

Toni Perković, Ivo Stančić, Luka Mališa, and Mario Čagalj

FESB, University of Split, Croatia
{toperkov,istancic,lmalisa,mcagalj}@fesb.hr

Abstract. We consider the classical problem of establishing *initial* security associations in wireless sensor networks. More specifically, we focus on pre-deployment phase in which sensor nodes have not yet been loaded with shared secrets or other forms of authentic information.

In this paper, we propose two novel *multichannel* protocols for initialization of large scale wireless sensor networks. The first protocol uses only secret key cryptography and is suitable for CPU-constrained sensor nodes. The second protocol is based on public key cryptography. Both protocols involve communication over a bidirectional radio channel and an unidirectional out-of-band *visible light channel*. A notable feature of the proposed "public key"-based key deployment protocol is that it is designed to be secure in a very strong attacker model, where an attacker can eavesdrop, jam and modify transmitted messages by adding his own message to both a radio and a visible light channel; the attacker however cannot disable the visible light communication channel. We show that many existing protocols that rely on the visible light channel are insecure in this strong adversary model.

We implemented the proposed protocols on the Meshnetics wireless sensor platform. The proposed protocols are cheap to implement, secure in the very strong attacker model, easy to use and scalable. We also designed and tested a simple random number generator suitable for sensor platforms.

1 Introduction

Deployment of cryptographic keys into individual sensor nodes is an imperative for secure operation of a sensor network. While there is a large body of work on key management in scenarios where cryptographic keys are already deployed into the nodes [9,11,12,21,27], very few studies exists on the equally important problem of establishing initial security associations in large wireless sensor networks.

Many existing systems consider the key pre-deployment to be a trivial matter. Thus, we can read that "the key distribution is relatively simple; nodes are loaded with a shared key before deployment". Long experience with WiFi networks have taught us that very often such "relatively simple" setup procedures render the security features useless (users easily give up and thus leave their networks unprotected), even when dealing with only a few network devices. Some other solutions propose to send the key in the clear over the radio channel or alternatively, imprint the keys onto the nodes at production time (ZigBee [1]). The problem with this approach is that customers may not trust the keys deployed by the factory.

Solutions that require physical contact are not scalable, especially if the user is required to initialize a large number of nodes. More advanced solutions have been proposed in [8,5,3,24,24,19,30] some of which do not scale well and/or require specialized node hardware, and some are insecure in the realistic attacker model introduced in this paper.

When dealing with initialization of network nodes on a large scale, a secure, fast, cost effective and above all user-friendly solution is mandatory. In this paper, we propose two novel *multichannel* protocols for initialization of large scale wireless sensor networks. Similar to [30], our protocols involve communication over a radio channel and the out-of-band visible light channel (VLC). The first protocol uses only secret key cryptography and is suitable for CPU-constrained sensor nodes. The "secret key"-based initialization of sensor nodes is depicted in Figure 1(a). In this protocol, each sensor node establishes a unique secret key with a *base station (BS)*. The base station comprises a simple web camera and one sensor node all attached to an ordinary PC. In the first phase of the protocol, the sensor nodes transmit secret keys to the base station over a *protected* visible light channel (Figure 1(a)). In the second phase, each sensor node runs a key verification protocol with the base station over a bidirectional radio channel. Once the keys are verified, the base station can serve as a *trusted third party* and mediate establishment of security associations between any pair or any group of sensor nodes.

Our second protocol uses public key cryptography. The "public key"-based sensor node initialization process is summarized in Figure 1(b). As with the previous protocol, the ultimate goal is to establish security association between each sensor node and the base station. This protocol is based on the multichannel pairing protocol from [38,5]. Thus, each sensor node first exchanges its public key (through specially formed commitment/openning pairs) with the base station over a radio channel (Figure 1(b)). In turn, each sensor node transmits a *short authentication string* (SAS) using a visible light channel (Figure 1(b) - right). The proposed "public key"-based protocol is similar to [30], with the difference that our protocol is designed to be secure in a very strong attacker model, where an attacker can eavesdrop, jam and modify transmitted messages by adding his own message to both a radio and a visible light channel; the attacker however cannot disable the visible light communication channel[1].

The paper is organized as follows: in Section 2 we state the problem and assumptions. In Sections 3 and 4 we present the "secret key"- and "public key"-based protocols (including security analysis of both protocols). We describe the implementa-

[1] It was brought to our attention recently that a similar approach has been suggested in [31]. The initialization method in [31], however, is developed for a weaker attacker model than the one we consider here.

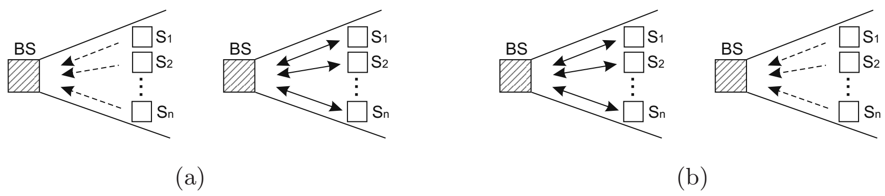


Fig. 1. Two phases of node initialization for (a) secret key and (b) public key deployment protocol. In (a) nodes transmit the key to the base station via VLC (dashed arrows) and perform authentication via a radio channel (full line arrows), while in (b) they exchange public keys over a radio channel and perform authentication via VLC.

tion of the protocols and a simple random number generator in Section 5. Related work is provided in Section 6. Finally, we conclude in Section 7.

2 Problem Statement and System Model

We consider the following problem: *How to securely initialize a large number of sensor nodes in a user-friendly fashion?* Since the initialization will be performed by potentially non-expert personnel, a solution has to be easy both to learn and use (*user-friendliness*). In addition, the hardware cost per node has to be minimized (*cost-efficiency*).

2.1 System Model

We assume that a user is equipped with a base station used for verification and monitoring as shown in Figure 2.

Base Station. The base station comprises a monitor, a simple web camera and one sensor node (a verification node) all attached to an ordinary PC. The verification node serves as a radio modem to the base station.

Uninitialized Sensor Nodes. Nodes may be equipped with a single LED (we used two LEDs in our implementation) used for key transmission via out-of-band VLC and with radio transceivers. In addition, each node has a "pushbutton" used to either restart or finalize the initialization process.

Cardboard box. A simple cardboard box is used to block the escape of light during the key transmission via VLC. The cardboard box is required only for the "secret key" - based key deployment protocol.

2.2 Attacker Model

An adversary has full control over the radio channel. He can eavesdrop, drop, delay, replay and modify messages sent via radio. Thus, he is able to initiate communication with any device (a node or the base station) and at any given

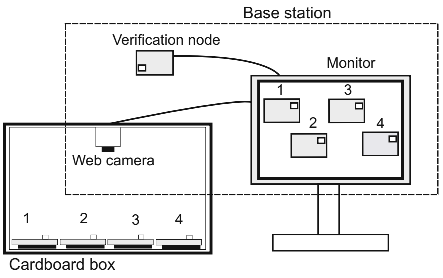


Fig. 2. Secret key deployment setup comprises a base station and a simple cardboard box

time during the key transmission. Furthermore, the adversary can install his own web camera in the same place where the initialization is taking place. We assume that devices involved in key deployment (PC and nodes themselves) are not compromised. Taking into account these constraints, we define: (a) *a passive adversary* who only observes the visual channel and can eventually record a secret key if the key transmission takes place in insecure conditions (outside the cardboard box), and (b) *an active adversary* who in addition can initiate communication with any device during the initialization phase.

In the case of "public key" - based initialization, we consider a stronger adversary model where an attacker can eavesdrop and modify messages sent over a light channel at all times (we elaborate this in Section 4).

3 Secret Key Deployment

In this section we propose secret key based key deployment protocol and provide initial security assessment of the proposed key deployment method.

3.1 Key Transmission and Verification

Prior to the start of node initialization, the user connects a web camera and a verification node to a PC. Next, the user places the web camera on top of the box from the inside, as shown in Figure 2. At this stage, the user turns the nodes ON and places them inside the box. Next, the user closes the box, runs the program on the PC and initiates the node initialization procedure. The box remains closed until the key transmission and verification is performed on all nodes which is subsequently indicated on the monitor.

Key transmission. Our "secret key"-based deployment is build upon ISO/IEV 9798-2 [4] three-pass key authentication protocol (Figure 3). We modify this protocol to include the communication over VLC (dashed arrow in Figure 3). The modified protocol evolves as follows.

The node S_i generates n-bit random key K_{S_iB} and k-bit random string N_{S_i}. The base station generates k-bit random string N_{B_i}. The node, equipped with

	Node S_i		Base Station BS
	Pick $K_{S_iB} \in_U \{0,1\}^n$		
	Pick $N_{S_i} \in_U \{0,1\}^k$		Pick $N_{B_i} \in_U \{0,1\}^k$
(1)		$\overset{S_i\|\|K_{S_iB}}{\dashrightarrow}$	
(2)		$\xleftarrow{B\|\|N_{B_i}}$	
(3)		$\xrightarrow{S_i\|\|\{N_{S_i}\|\|N_{B_i}\|\|B\}_{K_{S_iB}}}$	
			Verify N_{B_i}, K_{S_iB}
(4)		$\xleftarrow{B\|\|\{N_{S_i}\|\|N_{B_i}\}_{K_{S_iB}}}$	
(5)	Verify N_{S_i}, K_{S_iB}		

Fig. 3. Modification of ISO/IEV 9798-2 three pass key authentication protocol. The dashed arrow represents key transmission over secure VLC.

minimally one LED, sends the key via VLC (step 1) to the base station (web camera), as shown in Figure 3. At the same time, the base station performs three tasks: (i) collects keys K_{S_iB} generated by the nodes (step 1), (ii) initiates key verification over a radio channel (steps 2-5), and finally, (iii) notifies the user which node has been successfully initialized via the monitor. Section 5 provides details of the key transfer over VLC.

Key verification. After the key is transmitted over VLC, the base station initiates the key verification protocol. All messages in the key verification are exchanged over the radio channel. The base station (using the verification node) sends random nonce N_{B_i} over the radio channel to node S_i (step 2). Next, S_i forms a packet by encrypting concatenations $N_{S_i}\|N_{B_i}\|B$ with the key K_{S_iB}. The node S_i sends this message (and its identity) to the base station (step 3). The base station extracts the random nonce N_{S_i}, verifies the key K_{S_iB} and the random nonce N_{B_i}. If the verification is successful, the BS encrypts concatenation $N_{B_i}\|N_{S_i}$ using K_{S_iB} and sends it back to node S_i. The node S_i receives and verifies both the key K_{S_iB} and the random nonce N_{S_i}. The whole procedure is considered as completed if all the nodes are successfully initialized, which is finally indicated by the GUI on the monitor. At the end, the user opens the box and completes the initialization with the short push on the node's button. This feature is used to ensure the "proof of presence" property to prevent an active attack (as described in Section 3.3).

3.2 Sensor Node State Diagram

Both user and base station need to know the status of the initialization process at any given time. For that reason, the current state of the node will be indicated with a LED according to the state diagram shown in Figure 4. During the initialization process, the node can take one of the four following states: Uninitialized,

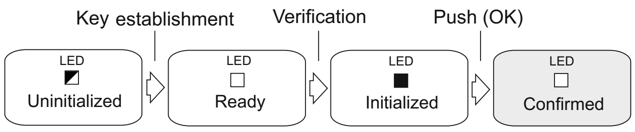


Fig. 4. Node's state diagram. A colored square indicates that the LED is ON, while a half colored that the LED is blinking.

Ready, Initialized and Confirmed. Next, we describe each of these states as well as the transitions between them.

Uninitialized state. Initially, when the user powers the node ON it is in the Uninitialized state. Prior to the start of key transmission the LED blinks with a predefined frequency. During this phase the node generates the key and, upon completion, sends it via VLC to the camera. After the key transmission is complete, the node advances to the Ready state (step 1 in Figure 3).

Ready state. The node remains in this state for a predefined period of time (e.g., a few seconds). In this state the node has sent the key and awaits the base station to initiate the key verification protocol over a radio channel. During this phase the node's LED is OFF (Figure 4). If the node does not receive any messages from the base station within the predefined period of time, it automatically restarts and returns back to the Uninitialized state. The node repeats the whole procedure which involves new key generation and transmission over VLC. Alternatively, the node receives a message from the base station and starts the key verification (steps 2-5 in Figure 2). If the key verification is successful, the node advances to the Initialized state.

Initialized state. In this state the node's LED is turned ON (Figure 4). At the same time, the base station notifies the user via the monitor about the node's position within the box as well as its current state. If the key verification succeeded on both sides (the node's and the base station's), the user is instructed to remove the nodes from the box and to shortly push the button on the node to finalize the initialization. The push of the button serves as "proof of presence", an aspect we describe in Section 3.3. However, if the node or the base station failed to verify the key, the user is instructed over the monitor to restart the initialization on selected nodes with a longer push on the button in order to repeat the node initialization. After the short push of the button, the node advances to the Confirmed state.

Confirmed state. In this state the node and the base station established a secret key and verified it, and the initialization process is finalized.

3.3 Initial Security Assessment

Our "secret key"-based protocol is build upon ISO/IEV 9798-2 [4] three-pass key authentication protocol that was proven to be secure when used over a radio channel. Therefore, we focus on possible attacks over the VLC, as we extended the ISO/IEV 9798-2 [4] protocol by including a transmission of a secret key via the VLC.

Camera recording (passive) attacker. A camera-recording attacker attempts to learn the secret key simply by recording the key sent from the node via VLC (step 1 in Figure 2). In this model the attacker does not interact in any way with the node initialization procedure.

Let us consider the case in which the node starts sending the key under insecure conditions (e.g., outside of the box). Thus, the attacker records the key, and the node advances from Uninitialized to Ready state (node's LED turns ON). In this state, the node waits a predefined period of time for the base station to initiate the key verification (Figure 4). After the predefined time period has passed during which the base station didn't initiate the key verification protocol, the node returns back to the Uninitialized state and repeats the whole procedure again (generates a new key and, again, sends it via VLC). The base station waits to receive a notification from the user that the system is ready (operates in *secure conditions*). Only then will the base station begin to process keys received over VLC and initiate the key verification protocol. Under secure conditions, the attacker does not have an access to the key transmitted by the node and therefore cannot successfully perform the key verification with the node.

Active attacker. In this attacker model, the attacker controls both the radio channel and communication over VLC when sensor node(s) are out of the cardboard box. Let us assume that the attacker captures the key sent by a node via VLC under insecure conditions (e.g., the node outside of the box). At this stage, the node is in Ready state and awaits the base station to initiate key verification (Figure 4). Next, the attacker initiates the key verification over the radio channel using the captured key. If the verification is successful on the node's side, the node advances to the Initialized state (the LED turns ON as shown in Figure 4). In this state the node waits for the user to confirm the initialization (push on a button). The user doesn't know that the attacker placed the node in the Initialized state so she picks the node up, and places it inside the box. Once the compromised node is placed inside the box, the base station recognizes a constantly powered ON LED on it and warns the user (via the monitor) to restart the initialization of that node. This is done by a longer push on the node's button. This form of active attacks does not work as the attacker does not have physical access to the node, therefore he cannot force the node to advance to Confirmed state. The user basically "proves her presence" through the push button.

4 Public Key Deployment

In this section we extend the attacker model to a more powerful adversary who can observe the electromagnetic radiation emanating from the LEDs. We assume the LEDs emanate radio signals which cannot be blocked by a simple cardboard box and we also assume that the attacker is able to easily eavesdrop on the leaked signals. This is a variant of an attack previously introduced in [18].

To establish keys between nodes and the base station by using a bidirectional radio channel and an unidirectional out-of-band VLC, we use SAS protocols [5,38]. The protocols make the key exchange process more usable, but at the cost of having to introduce public key cryptography. Recent work on elliptic curve cryptography has shown promising results regarding key distribution on resource constrained devices like our sensor nodes. In TinyPBC [26] and NanoECC [36] times less than 1 and 2 seconds, respectively, for point multiplication in binary fields were achieved.

Many prominent solutions that use LEDs and cameras [32,30] assume that the Visible Light Channel is authentic, which is not the case in our attacker model. To convey information via VLC they use on-off keying (switch the LED ON or OFF). An attacker equipped with a directional light source (e.g. a laser) has the capability to modify a message sent via VLC. In our model the attacker can modify messages by flipping $0 \rightarrow 1$, but not vice versa ($1 \rightarrow 0$) as the attacker cannot force a switched ON LED to turn OFF. In this case we speak of a *semi-authentic* visible light channel.

In the following sections we describe how to perpetrate such attacks and we also propose solutions on how to protect against them.

4.1 Attacks on Visible Light Channel

We consider prominent device pairing methods proposed in [32] and [30]. Both of the methods were developed for an authentic VLC (an attacker cannot modify messages sent via VLC). The proposed methods are secure within the authentic VLC model but, as we will show, are insecure in our semi-authentic VLC model (an attacker can flip $0 \rightarrow 1$).

Protocol [32] in the semi-authentic model. In [32] two devices (S_1 and S_2 as shown in Figure 5(a)) exchange public key values via a radio channel using the

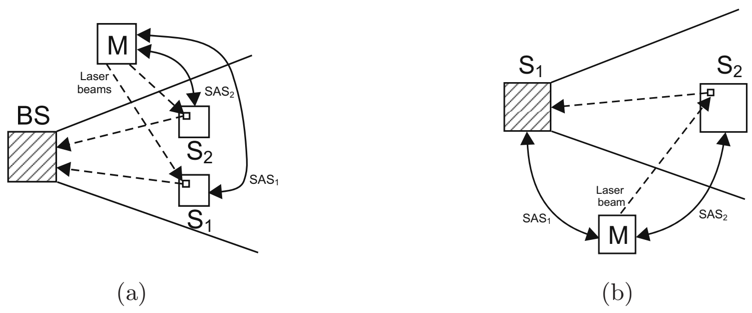


Fig. 5. Attacker M, with the aid of a laser, tries to modify short authentication strings exchanged over VLC (dashed arrows) between devices S_1, S_2 and BS (full-line arcs represent communication over a radio channel)

SAS protocol [5,38]. To authenticate these messages, each device simultaneously transmits short authentication strings (SAS) using visible light. The camera (BS in Figure 5(a)) captures both of these authentication strings and compares them. As BS does not know the SAS beforehand, the attacker can mount a MITM attack and modify these strings with a laser. Attacker M exchanges public keys with two devices S_1 and S_2 via a radio channel. When transmission via VLC occurs, the attacker points the lasers into the nodes' LEDs and appropriately modifies the bits (flips $0 \rightarrow 1$). The simplest attack is the one in which the attacker flips all bits $0 \rightarrow 1$. In this case, the base station will see all 1s and inform the user about the correct authentication. Please note that all 1s is a legitimate SAS.

Protocol [30] in the semi-authentic model. In an approach similar to [32], two devices (S_1 and S_2 in Figure 5(b)) exchange public key values over a radio channel. In this scheme, at least one device has an integrated web camera. In order to verify the exchanged public key values, device S_2 sends the SAS via VLC (using LEDs) to the device S_1. Here, an attacker tries to mount a MITM attack by exchanging different public keys with devices S_1 and S_2, (Figure 5(b)). To succeed, the attacker has to ensure that $\mathsf{SAS}_1 = \mathsf{SAS}_2$. Due to the property of the protocols [5] in which the probability for SAS_1 and SAS_2 to be equal is 2^{-k} (k is the length of the SAS) the attacker will establish two different authentication strings SAS_1 and SAS_2 with a high probability. However, in the semi-authentic model where the adversary can modify the bits (flip $0 \rightarrow 1$) this probability is significantly reduced. Indeed, if the ith bits of SAS_1 and SAS_2 are equal, an attacker will not need to modify them in any way. On the other hand, if the ith bits of SAS_1 and SAS_2 equal 1 and 0, respectively, an attacker could flip $0 \rightarrow 1$ by using the laser. Finally, if the ith bits of SAS_1 and SAS_2 are 0 and 1, the attacker will be unable to flip $1 \rightarrow 0$ for he cannot switch OFF an already powered ON LED. This is summarized below:

SAS_{1i}	SAS_{2i}	Attack
0	0	yes
0	1	no
1	0	yes
1	1	yes

Thus, we conclude that that 3 combinations of ith bits of SAS_1 and SAS_2 are beneficial for the attacker (all combinations but the second one). It follows that the probability for an attacker to modify the bits is $3/4$, therefore, the probability of a successful attack increases to $(3/4)^k$ as opposed to 2^{-k} (in the case of authentic VLC). If $k = 15$, the probability in a single attack increases from 2^{-15} to approximately 2^{-6}.

Virtual node attack. Let us assume the user wants to initialize one node (S_1) and the attacker (M) wants to inject his own virtual node (S_2) as shown in Figure 6. Attacker M simply exchanges public key values over a radio channel with BS and points his laser within the visible area of the base station's camera. The pointed laser is used to create a virtual node (device S_2 in Figure 6), and as

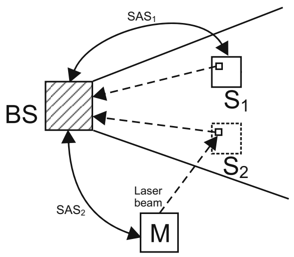


Fig. 6. An example of the virtual node attack; dashed arrows and full-line arcs represent communication over a semi-authentic VLC and a radio channel, respectively

such, to "blink" the correct short authentication string in such a way that the base station's camera detects it. The BS compares the SAS_2 it received from the attacker's laser with the one established over radio, sees that they match, and accepts the public key values from M as authentic.

4.2 "Public Key"–Based Deployment Protocol

We assume that each node S_i and the base station BS previously generated public key values pk_{S_i} and pk_B. In order to exchange authenticated public key values over a radio channel, we propose using the protocol introduced in [5,38], and shown in Figure 7. Please note, the base station performs this protocol individually with each node. The protocol evolves as follows:

(i) The user counts the number of nodes he/she wants to initialize and enters the number into the base station control software via a keyboard. We will show

$$
\begin{array}{rcl}
\text{Node } S_i & & \text{Base Station } BS \\
\text{Pick } N_{S_i} \epsilon_U \{0,1\}^k & & \text{Pick } N_{B_i} \epsilon_U \{0,1\}^k \\
m_{S_i} \leftarrow 1 \| S_i \| pk_{S_i} \| N_{S_i} & & m_{B_i} \leftarrow 0 || B || pk_B || N_{B_i} \\
(c_{S_i}, d_{S_i}) \leftarrow \text{commit}(m_{S_i}) & & (c_{B_i}, d_{B_i}) \leftarrow \text{commit}(m_{B_i}) \\
& \xleftarrow{\quad c_{B_i} \quad} & \\
& \xrightarrow{\quad c_{S_i} \quad} & \\
\widehat{m}_{B_i} \leftarrow \text{open}(\widehat{c}_{B_i}, \widehat{d}_{B_i}) & \xleftarrow{\quad d_{B_i} \quad} & \\
\mathsf{SAS}_{S_i} \leftarrow \ \widehat{N}_{B_i} \oplus N_{S_i} & \xrightarrow{\quad d_{S_i} \quad} & \widehat{m}_{S_i} \leftarrow \text{open}(\widehat{c}_{S_i}, \widehat{d}_{S_i}) \\
& & \mathsf{SAS}_{B_i} \leftarrow \ N_{B_i} \oplus \widehat{N}_{S_i} \\
& \overset{\mathsf{SAS}_{S_i}}{\dashrightarrow} & \text{Verify } \mathsf{SAS}_{S_i} = \mathsf{SAS}_{B_i}
\end{array}
$$

If $\mathsf{SAS}_{S_i} = \mathsf{SAS}_{B_i}$, the base station informs the user to accept public key values as authentic.

Fig. 7. SAS protocol by [5,38]. The dashed arrow represents communication over a semi-authentic VLC.

later that by entering the number of nodes we can prevent the virtual node attack and make the size of the SAS invariant of the number of nodes to be initialized.

(ii) The user switches the nodes ON and places them in front of the camera, with the LEDs facing the camera.

(iii) The node's LED starts flashing with the delimiter 111000 to indicate to the BS they are ready to be initialized and to enable the BS to count them.

(iv) Next, the user instructs the base station to begin with the protocol shown in Figure 7. Having exchanged commit/open pairs with the BS, each node S_i first calculates the respective SAS_{S_i} (Figure 7), Manchester encodes SAS_{S_i} and begins transmitting it repetitively via a VLC (using on-off keying, switching LED OFF and ON). The Manchester encoded short authentication string, denoted $M(\mathsf{SAS}_{S_i})$, is separated with delimiter 111000. The usage of the delimiter and Manchester encoding was inspired by I-codes [6] and was used to prevent the flipping attacks (Section 4.3). Finally, the node transmits (blinks) the following repetitive sequence:

$$\cdots\overbrace{1\,1\,1\,0\,0\,0}^{\text{delim.}}\ \overbrace{1\,0\,0\,1\cdots1\,0}^{M(\mathsf{SAS}_{S_i})}\ \overbrace{1\,1\,1\,0\,0\,0}^{\text{delim.}}\ \overbrace{1\,0\,0\,1\cdots1\,0}^{M(\mathsf{SAS}_{S_i})}\ \overbrace{1\,1\,1\,0\,0\,0}^{\text{delim.}}\cdots$$

(v) If the SAS verification is successful for **all** the nodes, the user is instructed to finalize the initialization procedure by pushing a button on each of the nodes. If one or more nodes fail to initialize properly (e.g. due to errors in transmission, attacks etc.) the initialization procedure is aborted for all the participating nodes.

4.3 Short Security Analysis

Due to the lack of space, in this section we provide only a short security analysis of the public key deployment protocol.

Flipping attacks. In order to prevent flipping attacks we used Manchester encoded SAS_{S_i} for the transmission via VLC. Note that such a message contains an equal number of 0s and 1s. Due to the on-off keying modulation and the fact that an attacker is unable to switch OFF the LED (flip $1 \to 0$), any attempt of flipping will be detected by the BS as an excess of 1s. This construction is proved secure in [6].

Virtual node attack. According to the protocol, the base station knows exactly how many nodes it has to initialize (step (i) of the protocol). In addition, the BS counts itself the nodes by detecting respective delimiters (111000) transmitted over VLC. In order to successfully inject his own virtual nodes, the attacker has to block transmission of the delimiter 111000 over VLC for at least one of the nodes. However, the attacker cannot do this, for he is unable to turn OFF an already switched ON LED. In addition, any attempt of flipping $0 \to 1$ in the delimiter will be detected by the BS [6].

All or none. The design choice to abort the initialization procedure if at least one node fails to initialize properly makes the SAS invariant to changing the number of nodes to be initialized. Indeed, from the above analysis we know that an attacker can neither add new (virtual) nodes, remove existing (legal) ones, nor perform bit flipping attacks. It follows that the attacker can only try to perform a man-in-the-middle attack against one or more legal nodes. Now, if an attacker attempts to mount the attack against m nodes (out of n) and the respective short authentication strings are mutually independent, the probability of a successful attack against at least one sensor node, in a single attempt, will be at least $\min\{m \cdot 2^{-k}, 1\}$ [5]. For example, if the attacker attacks $m = 100$ nodes and $k = 15$, the probability for the attacker to succeed against at least one node is around 2^{-8}. However, by restricting the attacker to be successful against all the nodes, the probability for the attacker to succeed is reduced to $(2^{-15})^m = 2^{-15 \cdot m}$. Therefore, the best strategy for the attacker is to mount an attack against exactly one node (i.e., $m = 1$), which implies the probability of success (in a single attempt) to be bounded by $2^{-k} + \varepsilon$ (k being the size of SAS and ε a negligible probability) [5].

5 Implementation

We next describe the implementation of our secret-key deployment protocol. More specifically, we describe the implementation of a simple random number generator (RNG) and the key recognition software that enables communication over the light channel. We used Meshnetics ZigBee sensor nodes equipped with Green and Red LEDs, Atmel AT-mega1281V microcontrollers and AT86RF230 RF transceivers. Each sensor module features 128KB of flash memory and 8KB of RAM with data rate of 250 kbps in frequency band from 2.400 – 2.483 GHz. For software developing and testing of the initialization procedure, a PC with the following configuration was used: Intel dual core processor clocked at 2.66GHz, 2GB of RAM, a Logitech notebook deluxe webcam with VGA resolution at 30fps interfaced via USB to the computer and Windows XP SP3 operating system.

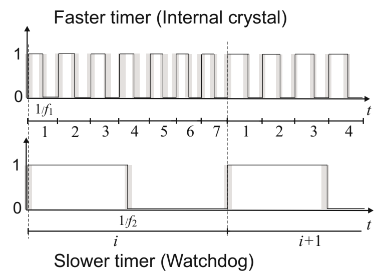
5.1 Random Number Generator

The key feature for secure communication lies in a good random number generator. In this section, we describe our Random Number Generator (RNG). We first describe some related work on random number generators suitable for devices with limited processing capabilities.

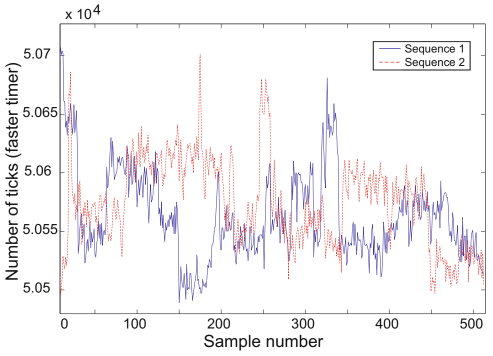
TinyRNG [15] uses transmission bit errors as a source of randomness. These bits are randomly distributed as well as uncorrelated and may not be manipulated by an adversary. In [14] two oscillators are used, one oscillating much faster than the other. Generated bit stream's randomness is based on the frequency instability of a free running oscillator. The slow oscillator samples the higher frequency oscillator. They have shown that if the jitter in the slow oscillator signal is sufficient, the output of the RNG will have very little bit-to-bit correlation. Tkacik [37] also uses two free-running oscillators whose frequency vary with voltage and temperature. Random numbers are generated as exclusive-or of previously selected and permuted 32 bits of the LFSR (linear feedback shift register) and CASR (cellular automata shift register). Each shift register is clocked by these oscillators. However, an initial seed is required for each register.

Design of a Random Number Generator. In our implementation we used the approach from [14]. The generation of random numbers goes as follows: Meshnetics ZigBee nodes are equipped with two usable oscillators, an Internal Calibrated RC Oscillator (4 MHz) and a Watchdog Oscillator (128 kHz) [25]. The software running on the sensor nodes creates two timers; one timer is associated with the slower oscillator and the other timer with the faster one. The timers are configured with clock dividers in such a way that the slower timer fires once per second, while the faster one fires roughly 50000 times per second. On every tick of the slower timer, the number of ticks from the faster timer is logged. Figure 8(b) shows two traces of the number of ticks from the faster timer during the period of 512 ticks from the slower timer (roughly 512 seconds). As shown, the source of randomness comes from the instability (jitter) of the two used oscillators (Figure 8(a)).

Digital postprocessing. Table 5.1 shows the digital postprocessing and the random number generation process. As shown, on each successful low frequency timer tick the number of high frequency ticks is counted. Next, this value is



(a)



(b)

Fig. 8. (a)An example of oscillator frequency instability (jitter). (b) Two traces showing the number of ticks of a faster timer relating to one tick of the slower one.

Table 1. An example of digital postprocessing performed on the generated raw bitstream as well as the generation of random numbers. The generated bit stream is 110100.

Number of ticks (Watchdog)	Number of ticks (Internal RC)	Partial binary representation	Last two digits
1	50607	10101111	11
2	50605	10101101	01
3	50640	11010000	00

converted into a binary representation (last eight binary digits are presented in Table 5.1) from which the last two bits are taken. We could extract more than two bits at the expense of a more complex extractor. Since for our purpose the entropy is sufficient, we choose to use this simple extractor. The results of statistical tests are presented in the next section.

Statistical Tests. ENT [40] and NIST [13] statistical test suites were used to test the randomness of our generated bitstreams. Statistical tests were conducted on a 3×10^6 long bitstream which we obtained from 7 ZigBee nodes over the period of approximately 3 days.

ENT [40] is a pseudorandom number frequency test that performs a variety of tests such as Entropy, Arithmetic mean, Monte Carlo value for Pi, Serial correlation coefficient and Chi square distribution. Table 5.1 contains the results of the ENT test performed on a 3×10^6 long bitstream.

Only the last two bits were taken from the binary representation of the faster timer tick count. If more than two bits are taken, the bitstream fails the “Chi square” part of the ENT test suite. But, as we already mentioned, our RNG directly samples the number of faster timer ticks, without the requirement for other complex extractors.

NIST STS [13] contains 15 tests out of which only 8 were performed due to the minimum bitstream requirement (3×10^6 bits were produced) for each test. Each test is used to calculate the P-value which shows the strength of the null hypothesis. The hypothesis passes the test if the P-value is higher than 0.01 in which case the sequence is considered to be random. As shown in Table 5.1, the generated sequence passed the tests (P-value is higher than 0.01).

Table 2. ENT test results

Entropy = 0.999999 bits per bit.
Optimum compression would reduce the size of this 3×10^6 bit file by 0 percent.
Chi square distribution for 3267632 samples is 2.40 and randomly would exceed this value 12.14 percent of the times.
Arithmetic mean value of data bits is 0.4996 (0.5 = random).
Monte Carlo value for Pi is 3.155460889 (error 0.44 percent).
Serial correlation coefficient is 0.000264 (totally uncorrelated = 0.0).

Table 3. NIST test results

TEST	P-VALUE	PROPORTION	TEST	P-VALUE	PROPORTION
frequency	0.148094	0.9922	fft	0.468595	1.0000
block-frequency	0.500934	0.9922	aperiodic	all passed	all passed
cumulative-sums	0.311542	0.9922	apen	0.275709	0.9844
cumulative-sums	0.031497	0.9922	serial	0.671779	0.9844
runs	0.437274	0.9922	serial	0.637119	0.9922

These tests were performed over the raw bits. Since the output bitstream passes both NIST and ENT test suites, no additional randomness extractors (universal hash functions [41,7], von Neumann extractor [39], or simply applying a cryptographic hash function over the bitstream) are necesarry.

These results are preliminary; future work will include a more detailed study of factors which impact the work of RC oscillators (e.g. voltage and temperature), and therefore directly impact the quality of the generated random numbers.

5.2 Communication over a Visible Light Channel

After the key generation follows the key transmission via an out-of-band Visible Light Channel (VLC). The sensor nodes are programmed in such a way that generated key bits are Manchester encoded prior to transmission which ensures lower bit error rates during the transmission over VLC. The bits are transmitted in such a way that logical 0 and 1 of our bitstream are represented with LED ON and OFF states, respectively. The duration of each state (single LED's blink) is approximately 200 ms. In Figure 9 we give an example of a bitstream's "life-cycle"; from the bit generation to the bit transmission phase. As shown in Figure 9, the generated bits are separated in such a way that the first and the second LED (Green and Red LED) transmit odd and even bits, respectively, of Manchester encoded binary stream via VLC. In this way we achieve easier key recognition on the side of the base station, as described in the sequel.

Computer Vision. Once the user places sensor nodes inside of the box, we use our computer vision (CV) system to derive the secret key from the nodes' LED blinking sequence. We developed our CV system in MATLAB 2007 GUI [16], and achieved transmission speeds of 10 bits per second (5 b/s per each LED).

The image processing part of our CV system is CPU demanding. In order to achieve real-time performance, we process only certain parts of an webcam-obtained image - so called "Areas of Interest" (small rectangles encompassing LEDs of each node). The algorithm was designed to work with two LEDs on each node (Green and Red LED). To determine the Area of Interest (AoI) for each node, which is the first step, a few seconds of buffered frames is required. Once the areas are determined, the rest of the algorithm is performed in real-time. All of the following steps are performed only over Areas of Interest. The rest of the image does not contain any relevant information, and thus is excluded from future processing.

Image transformation. In the second step, the selected image parts (AoIs) are converted from RGB to HSV color space, known to be more reliable for detecting

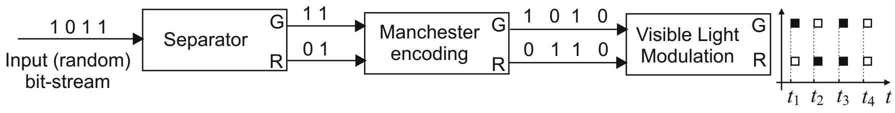


Fig. 9. An example of the bit stream sent via VLC using Manchester encoding. *G* and *R* stand for Green and Red LED, respectively.

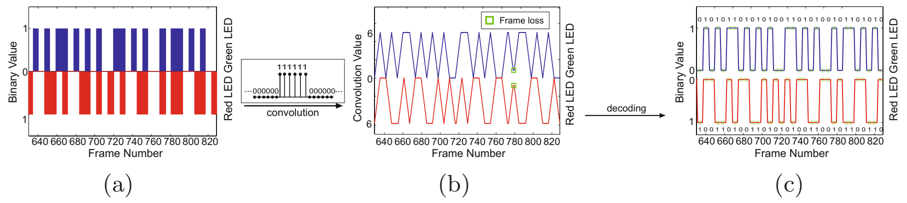


Fig. 10. A key recognition process: (a) detecting the status of LED indicators, (b) applying the convolution over the sampled area and (c) the bit identification process after the convolution

colors in low and changing light conditions [33]. Obtained images are tested for their levels of Hue, Saturation and Value, which enables us to detect the state (ON/OFF) of each LED. Color detector relies mainly on the level of Hue, while levels of Saturation and Value are just used in order to avoid false detection due to noise at low illumination conditions in the dark box.

Recognition of VLC signal. Due to a high frame loss and transmission error rates, during the transmission each bit is repeated in 6 consecutive frames (6 samples per bit). Decoding starts by detecting first 18 frames of the packet delimiter (3 binary ones in a sequence on both LEDs). Next, the key recognition algorithm performs the mathematical operation of convolution over the frames following the delimiter with a mask of six consecutive 1s (Figure 10(a)). As a result, data arrays containing values ranging from 0 to 6 are obtained (Figure 10(b)), where elements with extremes 0 and 6 are decoded as bits 0 and 1, respectively. Plateaus (areas with multiple, identical and consecutive elements) are decoded as double 0s or 1s, depending on their values (0 or 6). As Manchester encoding was used, only the convoluted signal's slope is analyzed, and not their values. This results in a method highly robust to de-synchronization effects. As shown by an example in Figure 10(b), frame loss during transmission via VLC does not affect correct bit recognition in any way.

6 Related Work

Recently, many key deployment schemes such as Zig Bee [1], SPINS [27], LEAP [42] and Transitory Master Key [10] have been proposed. Others [9,11,12,21,28] propose random key pre-distribution schemes. All of these schemes rely on an unspecified secure key deployment mechanism between devices.

In On-off Keying, the presence of an RF signal represents a binary 1, while its absence represents a binary 0 [5,6]. By using an unidirectional encoding scheme, On-off Keying ensures that an attacker is unable to modify a packet during transmission.

In Shake Them Up [8], user establishes a secret key between two nodes by holding and shaking the devices together while they send identical packets over the radio. This way, they assume that an adversary is unable to distinguish the source of the packets. This may be violated by using radio fingerprinting. Also, this does not scale well. The three related schemes are Are You With Me [20], Smart-Its Friends [17] and [22].

In Key Infection [2], two nodes establish a secret key by sending it in the clear over radio. They assume an attacker is unable to eavesdrop all the keys from all the nodes (e.g., 10.000 nodes) during key deployment. Based on simplicity and cost effectiveness, this scheme is insecure against a determined adversary. Moreover, an adversary is capable of injecting his own key, also violating key authentication.

In Resurrecting Duckling [35], a physical contact is required to securely establish a secret key. Based on the assumption that physical contact is secure, key authenticity and secrecy are ensured. But, since it requires specialized additional hardware, this scheme is not cost effective.

In Message In a Bottle [19], keys are sent in the clear to the nodes located inside a Faraday cage that ensures key secrecy and authenticity. However, the number of simultaneously initialized nodes determines the size of the Faraday cage. Moreover, a scale is used to determine the number of nodes within the Faraday cage based on total Faraday cage weight. In order to ensure key secrecy and authenticity for a large number of nodes, this scheme requires specialized setup hardware.

In HAPADEP [34] both data and verification information is sent over an audio channel. The pairing devices are both required to have speakers and microphones. In a related paper, Saxena and Uddin [30] present a device pairing method with an unidirectional channel based on devices equipped with LEDs and a video camera as the receiver. Their method is used for asymmetric pairing scenarios. Again, Saxena et. al. [32] use an auxiliary device (a laptop equipped with a web camera) to compare a short authentication string sent from the nodes to the laptop via unidirectional visible light channel. Both protocols are prone to laser attacks where an adversary may inject his/her malicious key by modifying the messages sent via the light channel with a directional light source (e.g. laser emitter).

Talking to strangers [3] requires specialized setup hardware (e.g. audio or infrared) in order to setup a public key. Seeing Is Believing uses an installation device with a camera or a bar code reader to create an out-of-band secure channel [24]. Key authenticity is achieved through certified public keys.

Mayrhofer and Welch [23] use an out-of-band laser channel constructed with off the shelf components for transmitting short authentication strings. According to [23], the proposed solution does not ensure complete authenticity of the the laser channel. Roman and Lopez [29] discuss general aspects of communication over a visible light channel.

7 Conclusion

We made several contributions in this paper. We proposed two novel *multichannel* protocols for initialization of large scale wireless sensor networks. The first protocol uses only secret key cryptography and is suitable for CPU-constrained sensor nodes. The second protocol is based on public key cryptography. Both protocols involve communication over a bidirectional radio channel and an unidirectional out-of-band *visible light channel*.

We demonstrated the importance of considering a very strong and realistic attacker model, where an attacker can eavesdrop, jam and modify transmitted messages in both a radio and a visible light channel; many existing protocols that rely on a visible light channel were shown to be insecure in this strong adversary model. Our "public key" - based protocol is designed to be secure in this very strong attacker model. Moreover, we showed that principle "all or none" keeps invariant the size of *short authentication strings* to changing the number of sensor nodes to be initialized.

The proposed protocols are implemented on the Meshnetics ZigBee sensor nodes. We showed that the proposed protocols are cheap to implement (a sensor node has to be equipped with one LED and a "push button") and scalable. We also designed and tested a simple random number generator suitable for CPU-constrained sensor nodes.

Acknowledgment

The authors would like to thank the anonymous reviewers for their thorough reviews and helpful suggestions.

References

1. ZigBee Alliance. ZigBee Specification (Document 053474r06, Version 1.0). Technical report (June 2005)
2. Anderson, R., Chan, H., Perrig, A.: Key Infection: Smart Trust for Smart Dust. In: IEEE International Conference on Network Protocols (2004)
3. Balfanz, D., Smetters, D.K., Stewart, P., Wong, H.C.: Talking to Strangers: Authentication in Ad-hoc Wireless Networks. In: Symposium on Network and Distributed Systems Security (2002)
4. Boyd, C., Mathuria, A.: Protocols for Authentication and Key Establishment. Springer, Heidelberg (2003)
5. Cagalj, M., Capkun, S., Hubaux, J.: Key Agreement in Peer-to-Peer Wireless Networks. In: Proceedings of the IEEE Special Issue on Cryptography and Security (2006)
6. Cagalj, M., Hubaux, J.P., Capkun, S., Rengaswamy, R., Tsigkogiannis, I., Srivastava, M.: Integrity (I) Codes: Message Integrity Protection and Authentication Over Insecure Channels. In: Proceedings of the IEEE Symposium on Security and Privacy (2006)
7. Carter, L., Wegman, M.N.: Universal Classes of Hash Functions. Journal of Computer and System Sciences 18(2) (1979)

8. Castelluccia, C., Mutaf, P.: Shake Them Up!: A Movement-based Pairing Protocol for CPU-constrained Devices. In: ACM MobiSys (2005)
9. Chan, H., Perrig, A., Song, D.: Random Key Predistribution Schemes for Sensor Networks. In: Proceedings of the IEEE Symposium on Security and Privacy (2003)
10. Deng, J., Hartung, C., Han, R., Mishra, S.: A Practical Study of Transitory Master Key Establishment For Wireless Sensor Networks. In: Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks (2005)
11. Du, W., Deng, J., Han, Y.S., Varshney, P.K.: A Pairwise Key Pre-Distribution Scheme for Wireless Sensor Networks. In: Proceedings of the 10th ACM conference on Computer and Communications Security, CCS (2003)
12. Eschenauer, L., Gligor, V.D.: A Key-Management Scheme for Distributed Sensor Networks. In: Proceedings of the 9th ACM conference on Computer and Communications Security (2002)
13. Rukhin, A., et al.: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications (2001), `http://csrc.nist.gov/rng/`
14. Fairfield, R.C., Mortenson, R.L., Coulthart, K.B.: An LSI random number generator (RNG). In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 203–230. Springer, Heidelberg (1985)
15. Francillon, A., Castelluccia, C.: TinyRNG: A Cryptographic Random Number Generator for Wireless Sensors Network Nodes. In: Int. Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (2007)
16. MATLAB Online Users Guide, `http://www.mathworks.com` (last access, September 2008)
17. Holmquist, L.E., Mattern, F., Schiele, B., Alahuhta, P., Beigl, M., Gellersen, H.W.: Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts. In: International Proceedings of the 3rd international conference on Ubiquitous Computing (2001)
18. Kuhn, M.G.: Electromagnetic eavesdropping risks of flat-panel displays. In: Martin, D., Serjantov, A. (eds.) PET 2004. LNCS, vol. 3424, pp. 88–107. Springer, Heidelberg (2005)
19. Kuo, C., Luk, M., Negi, R., Perrig, A.: Message-In-a-Bottle: User-Friendly and Secure Key Deployment for Sensor Nodes. In: ACM SenSys (2007)
20. Lester, J., Hannaford, B., Borriello, G.: "Are you with me?" - using accelerometers to determine if two devices are carried by the same person. In: Ferscha, A., Mattern, F. (eds.) PERVASIVE 2004. LNCS, vol. 3001, pp. 33–50. Springer, Heidelberg (2004)
21. Liu, D., Ning, P., Du., W.: Group-Based Key Pre-Distribution in Wireless Sensor Networks. In: ACM Workshop on Wireless Security (2005)
22. Mayrhofer, R., Gellersen, H.: Shake Well Before Use: Two Implementations for Implicit Context Authentication. In: Ubicomp (2007)
23. Mayrhofer, R., Welch, M.: A Human-Verifiable Authentication Protocol Using Visible Laser Light. In: International Conference on Availability, Reliability and Security (2007)
24. McCune, J.M., Perrig, A., Reiter, M.K.: Seeing-Is-Believing: Using Camera Phones for Human-Verifiable Authentication. In: Proceedings of the IEEE Symposium on Security and Privacy (2005)
25. Murray, K.D.: 8-bit AVR Microcontroller with 64K/128K/256K Bytes In-System Programmable Flash, `http://www.atmel.com` (last access, March 2008)

26. Oliveira, L.B., Scott, M., Lopez, J., Dahab, R.: TinyPBC: Pairings for Authenticated Identity-Based Non-Interactive Key Distribution in Sensor Networks. In: 5th International Conference on Networked Sensing Systems, INSS (2008)
27. Perrig, A., Szewczyk, R., Tygar, J.D., Wen, V., Culler, D.E.: SPINS: Security Protocols for Sensor Networks. Wireless Networks 8(5) (2002)
28. Ramkumar, M., Memon, N.: An Efficient Key Predistribution Scheme for Ad-hoc Network Security. IEEE Journal on Selected Areas in Communications (2005)
29. Roman, R., Lopez, J.: KeyLED - Transmitting Sensitive Data Over Out-of-Band Channels in Wireless Sensor Networks. In: IEEE WSNS (2008)
30. Saxena, N., Uddin, M. B.: Automated Device Pairing for Asymmetric Pairing Scenarios. In: Chen, L., Ryan, M.D., Wang, G. (eds.) ICICS 2008. LNCS, vol. 5308, pp. 311–327. Springer, Heidelberg (2008)
31. Saxena, N., Uddin, M.B.: Bootstrapping Key Pre-Distribution: Secure, Scalable and User-Friendly Initialization of Sensor Nodes. In: ACNS (2009)
32. Saxena, N., Uddin, M.B., Voris, J.: Universal Device Pairing Using an Auxiliary Device. In: Proceedings of the 4th Symposium on Usable Privacy and Security, SOUPS (2008)
33. Shapiro, G., Stockman, G.C.: Computer Vision. Prentice-Hall, Englewood Cliffs (2001)
34. Soriente, C., Tsudik, G., Uzun, E.: HAPADEP: Human-Assisted Pure Audio Device Pairing. In: Proceedings of the 11th International Conference on Information Security, ISC (2008)
35. Stajano, F., Anderson, R.: The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. In: 7th International Workshop. Springer, Heidelberg (1999)
36. Szczechowiak, P., Oliveira, L.B., Scott, M., Collier, M., Dahab, R.: NanoECC: Testing the Limits of Elliptic Curve Cryptography in Sensor Networks. In: Verdone, R. (ed.) EWSN 2008. LNCS, vol. 4913, pp. 305–320. Springer, Heidelberg (2008)
37. Tkacik, T.E.: A Hardware Random Number Generator. Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems, CHES (2003)
38. Vaudenay, S.: Secure Communications Over Insecure Channels Based on Short Authenticated Strings. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 309–326. Springer, Heidelberg (2005)
39. von Neumann, J.: Various Techniques Used in Connection With Random Digits. Applied Math Series (1951)
40. Walker, J.: Hotbits, `http://www.fourmilab.ch/random/` (last access, March 2009)
41. Yuksel, K., Kaps, J.P., Sunar, B.: Universal Hash Functions for Emerging Ultra-Lowpower Networks. In: Proceedings of the Communications Networks and Distributed Systems Modeling and Simulation Conference (2004)
42. Zhu, S., Setia, S., Jajodia, S.: LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks. In: Proceedings of the 10th ACM conference on Computer and Communications Security, CCS (2003)