Key Distribution for Wireless Sensor Networks and Physical Unclonable Functions*

Jorge Guajardo, Sandeep S. Kumar, and Pim Tuyls

Philips Research Europe, Eindhoven, The Netherlands
{jorge.guajardo,sandeep.kumar,pim.tuyls,}@philips.com

Abstract. Virtually all applications which provide or require a security service need a secret key. In an ambient world, where (potentially) sensitive information is continually being gathered about us, it is critical that those keys be both securely deployed and safeguarded from compromise. In this paper, we provide solutions for secure key deployment in sensor networks based on the use of Physical Unclonable Functions (PUFs). In particular, we show how the properties of Fuzzy Extractors or Helper Data algorithms can be used to securely deploy secret keys to a low cost wireless node. Our protocols are more efficient (round complexity) and allow for lower costs compared to previously proposed ones.

Key Words. Physical Unclonable Functions, Fuzzy Extractor, Helper Data Algorithm, Sensor Nodes, Key Distribution.

1 Introduction

It is natural that all security and privacy preserving protocols use some sort of secret-key material regardless of whether the protocols are based on public-key or private-key cryptography. The interesting fact, however, is that everyone assumes that the key is magically deployed onto the nodes of the network in a safe and secure manner as noted most recently by Kuo et al. in [21]. Most notably, one of the best examples of sensor node deployment in the "real world," the ZigBee specification [35], assumes that either the nodes will be loaded with their key material by sending the keys in the clear (resulting in a brief vulnerability window) or that factory initialized keys are pre-loaded on the nodes. Kuo et al. [21] notice, however, that such factory pre-set keys might not be trusted by many users. In this paper, we propose a new method for secure key deployment of sensor node keys based on the properties of Physical Unclonable Functions (PUFs) and fuzzy extractor schemes. The advantages range from the added security guarantees provided by tamper evidence, tamper resistance and unclonability as provided by PUFs, to somewhat simplified protocols for the end-user (the individual deploying a wireless sensor network). In addition, we show that under relaxed (but reasonable) security assumptions we can provide (i) costs reduction, since our protocols do not require additional hardware set-up devices as the Message-In-a-Bottle (MIB) protocol [21] does and (ii) considerably simplified protocols. Notice that we choose to compare to the MIB protocol since, to our knowledge, it is the only protocol that has thoroughly considered all requirements that must be satisfied by a key deployment protocol.

2 PUFs and Helper Data Schemes

In 2001, Pappu et al. [27] introduced the concept of Physical Random Functions or Physical Unclonable Functions. Here a physical object or device is used to define a function. In particular, upon challenging such a PUF with a challenge C_i, a response R_i is generated. Thus, we write: $R_i \leftarrow \text{PUF}(C_i)$. PUFs have essentially two parts: i) a physical part and ii) an operational part. The physical part is a physical system that is very difficult to clone[1]. It inherits its unclonability from uncontrollable process variations during

* Presented at the Secure Component and System Identification Workshop - SECSI, March 17-18, 2008, Berlin, Germany

[1] Note that this stands in sharp contrast to Quantum Cryptography where cloning is impossible due to the basic laws of nature. In the case of PUFs, there is a very small (but non-zero) probability that the structure can be cloned.

manufacturing. In the case of PUFs on an IC such process variations are typically deep-submicron variations such as doping variations in transistors. The operational part corresponds to the function. In order to turn the physical system into a *function* useful for identification, a set of challenges C_i (stimuli) has to be available to which the system responds with a set of sufficiently different responses R_i. Examples of PUFs include optical PUFs [27], silicon PUFs [15] and coating PUFs [32]. In [16] the notion of an *Intrinsic* PUF or IPUF (a PUF inherently present in a device) was introduced targeting FPGAs. In [18], a similar idea is presented on an ultra-low power chip used in sensor node applications. Recently, [33] has introduced ultra-low cost identifiers based on randomized LC-circuits.

PUF Security Properties. As in any security system, in order to evaluate the security of the system, it is necessary that we state the necessary assumptions for the system to be secure. Previous works [27, 15, 32, 17, 16] have either explicitly or implicitly made the following assumptions: (i) It is assumed that a response R_i (to a challenge C_i) gives only a small amount of information on another response R_j (to a different challenge C_j) with $i \neq j$ and (ii) Without having the corresponding PUF (i.e. the actual physical device or structure) at hand, it is impossible to come up with the response R_i corresponding to a challenge C_i, except with negligible probability. In most cases, it is also reasonable to assume that PUFs are tamper evident. This implies that when an attacker tries to investigate the PUF to obtain detailed information about its structure, the PUF is damaged and the challenge-response behavior is changed substantially.

Fuzzy Extractor and Helper Data Schemes. Since PUF responses are noisy and the responses are not fully random, a Fuzzy Extractor or Helper Data algorithm is required to extract secure keys from the PUF responses. For formal definitions of Fuzzy Extractors and Helper Data algorithms we refer to [12, 23]. Informally, we need to implement two basic primitives: (i) *Information Reconciliation* or error correction and (ii) *Privacy Amplification* or randomness extraction. In order to implement those two primitives, helper data W are generated during the *enrollment phase* and procedures Gen and Rep are run. In order to implement the procedures Gen and Rep an error correction code $\mathcal{C}$ and a set $\mathcal{H}$ of universal hash functions [7] is required. The Gen-procedure takes as input a PUF response(s) R and produces as output a key K and helper data $W = (W_1, W_2)$. This is achieved as follows. First, a code word $C_S \leftarrow \mathcal{C}$ is chosen at random from $\mathcal{C}$. Then, a first helper data vector equal to $W_1 = C_S \oplus R$ is generated. Furthermore, a hash function h_i is chosen at random from $\mathcal{H}$ and the key K is defined as $K \leftarrow h_i(R)$. The helper data W_2 is set to i. During the key reconstruction phase the procedure Rep is run. It takes as input a noisy response R' from the same PUF and helper data W and reconstructs the key K *i.e.* $K \leftarrow \mathsf{Rep}(R', W)$. This is accomplished according to the following steps: (1) *Information Reconciliation*: Using the helper data W_1, $W_1 \oplus R'$ is computed. Then, the decoding algorithm of $\mathcal{C}$ is used to obtain C_S. From C_S, R is reconstructed as $R = W_1 \oplus C_S$; and (2) *Privacy amplification*: The helper data W_2 is used to choose the correct hash function $h_i \in \mathcal{H}$ and to reconstruct the key as $K = h_i(R)$. The security of such constructions has been established in [20, 23, 12, 5].

3 Secure Key Deployment for Sensor Nodes with PUFs

As previously mentioned and noted recently in [21], many protocols assume some secure way of transferring an initial key to a wireless device without actually specifying how to accomplish such a task. The starting point for the protocols that we propose in this paper is the Message-In-a-Bottle (MIB) protocol of [21] as this is, to our knowledge, the only work that has actually addressed the problem in a thorough manner (see also Sect. 3.4). As we will see a basic building block in the protocol is a Faraday cage which provides privacy from eavesdroppers during the initial key set-up. We propose to depart from this approach and to use PUFs and the corresponding helper data algorithm as a secure manner to initialize the sensor node. These protocols will be explained in detail in Sect. 3.3. Before continuing, we summarize the attacker model and services offered by the MIB protocol. This will allow us to make a fair comparison later in Sect. 3.5.

3.1 Assumptions and Strong Attacker Model

The problem at hand is best explained by the example provided in [21]: a customer receives a shipment of new sensor nodes and using a wireless communication channel (no other channel is assumed because of cost considerations) he wants to set a shared secret between the uninitialized nodes and a wireless base station. Kuo et al. [21] provide a list of properties that a solution for this problem should provide: (1) **key secrecy:** an attacker has negligible chance of compromising the shared secret between nodes and base station; (2) **key authenticity:** an uninitialized node receives the key that the base station originally sent and not a key coming from an adversary; (3) **forward secrecy:** compromising one node does not compromise the keys on previously deployed nodes. More importantly, the attacker that compromises a node only gets knowledge to the current key and knows nothing of previously generated keys; (4) **demonstrative identification:** users physically handle devices in such a way that they are certain of which devices are communicating; (5) **robust to user error:** the system should be designed around users and for normal users (not expert cryptographers or security engineers). In addition, a user error should not result in key compromise; (6) **cost effective:** the proposed solution should not add to the costs of the sensor node and/or of the network; (7) **no public-key cryptography:** in general public-key (PK) cryptography implementations are more expensive in terms of program space, slower speed and, if implemented at the hardware level, silicon area. We add an extra property to this list that our solution would provide: **tamper evidence:** an adversary's ability to tamper with the node along the distribution channel.

As in [21], we also assume that installation personnel can be trusted and that they can follow simple instructions (as in a cooking recipe). Similarly, we assume that once an initial key has been set up, the nodes will use secure communication protocols. The work of [21] also assumes that other devices (keying device and keying beacon) are present to facilitate the key deployment. We will show that the required number of devices used during key deployment is reduced in one of our protocols compared to MIB and thus, that our solution is more cost efficient. In our solution, we also assume that there is a PUF present in the sensor node with its corresponding security properties. Notice that the presence of such a PUF does not necessarily increase the cost of the node as Intrinsic-PUFs are inherently present in silicon devices as shown in [16, 18]. We would in additional require a Fuzzy extractor implemented on the node. In any event, we expect the costs to be minimal. Finally, we assume a very powerful adversary, whose aim is to compromise the keys to be shared by the nodes and the base station. The attacker can overhear, intercept, and inject any messages into the communication channel. In this model, we also assume (as in [21]) that the attacker is omni-present, i.e. the attacker is present before, during, and after key deployment. Later, we relax the model (by making the attacker somewhat weaker) and show that such relaxation induces savings in terms of protocol complexity and hardware devices required for key deployment.

3.2 The Message-In-a-Bottle Secure Key Deployment Protocol

We describe in some detail the Message-In-a-Bottle key deployment protocol [21], as our new protocol can be seen as a modification of some sub-protocols in MIB. In MIB five different parties participate: (i) the base station (S), which controls the entire network and has the capabilities of a regular PC. The base station delegates key deployment to less powerful devices (the keying device and the keying beacon) by transferring the keying material via a secure channel such as a secondary physical USB interface; (ii) the sensor node (M), which is to share a secret with the base station. Upon being powered on or reset the node is in an uninitialized state, once the node receives the correct key, the node changes to an initialized state. Finally, if the key deployment fails, the node is in a rejected state without sharing a valid key with the base station; (iii) the keying device (D), which is placed inside a Faraday cage together with the node and sends the initial key information to the node when the Faraday cage is closed. Then, the node uses the keying information received from the keying device to derive the key; (iv) the keying beacon (B), which is used to signal that the Faraday cage is closed and jam the communication channel (outside the Faraday cage), thus preventing an eavesdropper from obtaining information leaked by the Faraday cage. In addition, the keying beacon provides the user with status information about the deployment and its outcome; (v) the user, who wants to perform the key deployment. The protocol assumes weak time synchronization between D and B. This is achieved via counters and authenticated messages between the devices. The authentication protocol uses a keyed Message-Authentication-Code (MAC). The key K_D

used to derive the deployment key for node M is generated by the base station and securely transmitted to D. Then, for every new node M, the following steps are followed:

1. Once D and B have exchanged authenticated synchronization messages, the user turns on the node M and places D and M inside a Faraday cage. The user then closes the Faraday cage, the keying beacon is outside the Faraday cage and unable to communicate with D.
2. Inside the Faraday cage, D generates the node's M key as a pseudo-random function of the node's ID M, keyed with the current value of K_D, i.e. $K_M = \mathrm{PRF}_{K_D}(M)$. The keying device updates $K_D \leftarrow \mathsf{Hash}(K_D)$, increases a counter c by one (this is used to keep track of how many times K_D has been updated), computes $h \leftarrow \mathsf{Hash}(K_M)$ and sends h to M. The value h works as a commitment to K_M, which M can use later to verify validity of K_M. Notice that updating K_D via hashing ensures that a new key is used for every new node M and provides forward secrecy
3. D generates s random nonces $r_1, r_2, ..., r_s$, computes the activation key $k = K_M \oplus r_1 \oplus \cdots \oplus r_s$ and sends the r_i's to M over s rounds of communication along with the counter c. Thus, an attacker must overhear all s messages in order to compromise the key K_M.

In addition to the above mentioned tasks, the keying device D monitors the amount of noise in the background. If the Faraday cage is left open or it is not attenuating signals as expected, D will detect the presence of the keying beacon and abort the deployment. At the same time, the keying beacon jams all communications in the frequency of the deployment during the few seconds that the key deployment takes place. After such a short period of time, B signals the user to open the Faraday cage, and once the keying beacon and keying device verify that the protocol was performed as expected, the keying device sends the value of k (the validation key) to M, which then computes K_M and verifies that h (received in Step 2 of the protocol) corresponds to $\mathsf{Hash}(K_M)$. The end of the protocol verifies the correctness of the deployed keys by computing a MAC on the value k keyed with K_M. We refer to [21] for the details as they are not relevant to the discussion here.

The MIB protocol ensures that any user errors (like an open Faraday cage or too early opening) only leads to an erroneous key rather than to key leakage. This method requires no additional hardware per sensor node. However, it does require additional specialized hardware: *Keying Device*, *Keying Beacon* and a Faraday cage. Both, ease of use and robustness are achieved in the MIB protocol thanks to the ability of users to physically manipulate devices in such a way that they are certain which devices are communicating.

3.3 PUFs, Fuzzy Extractors and Their Use for Key Deployment

In this section, we present two protocols allowing secure key deployment to uninitialized sensor nodes. We make a distinction between two situations. The first protocol is similar in nature to MIB but we modify the key activation part at the end of MIB by using helper data as the activation key. We notice that the use of a secure area somewhere in the overall protocol seems to be a must. In other words, unless there is at some point in time an area in which the attacker can not eavesdrop, preserving key confidentiality seems unattainable. Thus, in the first protocol, we also make use of a Faraday cage. Notice that the first protocol provides the same guarantees as the MIB protocol, with reduce communication complexity and the added advantages of using a PUF. In other words, a PUF provides tamper evidence, unclonability and (depending on the PUF) tamper resistance.

In the second protocol, we assume the existence of a Trusted Third Party (TTP). This can be the manufacturer or a different entity charged with the authority of distributing and managing keys. We also show ways in which trust on the TTP in a "real-world" scenario (i.e. a world where an attacker can not be present everywhere) can be reduced to preserve the confidentiality of communications between sensor nodes and base station. This implies that in the second protocol we assume a weaker attacker model. In particular, the TTP has knowledge of the key and the hardware manufacturer can gain knowledge of the key if it is present both during the enrollment phase and during key deployment eavesdropping at the end-user premises. In such a weaker security model, we show that the use of PUFs allow for a significant reduction in the protocol complexity and a significant reduction in the hardware required for deployment. In addition, as in the first protocol, using PUFs provides unclonability, tamper evidence and tamper resistance, as well.

Secure Deployment without TTPs in The Strong Attacker Model. To achieve secure key deployment without a trusted third party, we require the same hardware that the MIB protocol requires: the base station (S), the keying device (D), the keying beacon (B), and the sensor node (M). In this case the hardware manufacturer has no involvement in the protocol. The protocol is shown in Fig. 2 in the appendix. We present the overall protocol for completeness. The set-up phase of the protocol is essentially the same as in MIB. The mutual authentication between the keying device and the keying beacon guarantees that both devices are not subject to a man-in-the-middle-attack[2] and the timestamps are used for weak synchronization between the devices. This way during the key activation phase of the protocol, both D and B can check that indeed the keying beacon was jamming the communication channel during key deployment and that the Faraday cage was closed.

The key deployment protocol is essentially performing the enrollment protocol as in Sect. 2. In addition, it stores a hash of the key and the helper data in M, which during the key validation and verification phase can be used by the node to check the validity of the activated key. The activation protocol is similar to the TTP-based protocol, i.e., the helper data W_M is sent over to the node (in the clear) and the node then constructs the key K_M. Notice that no information about the key is disclosed by sending W_M in the clear thanks to the fuzzy extractor constructions. During the verification phase the node M checks the validity of the hash value received during the key-deployment phase and proves to the base station that it is in possession of a valid key following the key verification protocol outlined in Fig. 2. Notice that in the MIB protocol security is somewhat enhanced by splitting the key into shares and transmitting the key shares over an extended period of time. This forces an adversary to be able to obtain all shares to successfully compromise the key. Similar techniques can be applied in our protocols if deemed necessary. In particular, instead of sending R_M in a single message, M could do this by computing $R'_M = R_M \oplus r_1 \oplus \cdots \oplus r_s$ and send the values $R'_M, r_1, \ldots, r_s$ one after another. This, however, would require the presence of a random number generator in the sensor M. An advantage in the current protocol is that the node does not require the presence of a random number generator to check the validity of its key. This translates into more space for performing other tasks, storing additional application code, or reduced hardware costs. The security of such scheme can be further enhanced by making the transfer of shares in a time delayed manner (see [4] for a description of a similar scheme in the RFID context). Such scheme has the advantage of not requiring a random number generator in M.

Secure Deployment with TTPs in a Weaker Attacker Model. In this protocol we assume the existence of a TTP. The protocol begins with a trusted third party performing an enrollment protocol by running $(K_M; W_M) \leftarrow \mathsf{Gen}(R_M)$ on the PUF response R_M as explained in Sect. 2. Observe that the TTP can be the hardware manufacturer (HWM) of the sensor nodes itself or an independent third party. The advantage of having an independent TTP is that the key is only known to the TTP and the end-user and not to the manufacturer. This is true since both R_M and W_M are necessary to reconstruct the correct key K_M and the manufacturer only knows R_M. Thus, we assume implicitly that the hardware manufacturer is not omnipresent. In particular, if desired the HWM could eavesdrop the deployment of the helper data W_M during key verification, thus gaining knowledge of the key K_M. This implies that we are working in a weaker attacker model (or alternatively, we trust the HWM). Notice, however, that this weaker attacker model provides us with significant reductions in both protocol complexity and hardware resources (i.e. cost) when compared to the original MIB protocol.

The values $(K_M; W_M)$ corresponding to node M are then sent to the user via a secure and authenticated channel. When the user receives the node and associated $(K_M; W_M)$ values, these are installed in the base station as corresponding to node M. Then, the following steps are performed:

1. The base station sends in the clear the value W_M to the node M.
2. Node M measures the PUF and obtains a response R'_M. Then, node M performs the information reconciliation and privacy amplification procedures, thus reconstructing the key $K_M \leftarrow \mathsf{Rep}(R'_M, W_M)$. Notice that the helper data has the same function as the activation key k in MIB.

[2] As in [3], we do *not* consider it to be an attack if the adversary only relays messages between the intended parties as this can not be prevented. In this case, (as noted in [3]) the adversary is simply acting as a wire. Thus, a man-in-the-middle attack requires modification of the messages as well.

3. The base station and the node then engage in a mutual authentication protocol such as the one suggested in [21] to verify the correctness of the installed key. Any other challenge-response protocol for mutual authentication can be used as well.

The overall protocol is shown in Fig. 1 in the appendix. We show an instantiation of the key verification part of the protocol based on standard mutual authentication techniques based on symmetric encryption. Notice, however, that similar protocols exist, which are based on Message-Authentication-Codes (MACs) or keyed hash functions.

It is clear that one disadvantage of the protocol is that the TTP knows the deployed key for the specific node M. However, this might be outweighed by the fact that our protocols do not require any additional hardware (i.e. no key beacon or keying device are required) and the protocol can be performed without having to introduce the nodes into a Faraday cage. Notice that the Faraday cage is still present at the manufacturer's side. However, in our protocol, it is the manufacturer or the TTP who have to invest in such secure facility, which we consider plausible. The question of tampering with the device during transit (between the manufacturer and the end-user) is also of no concern since PUFs guarantee tamper evidence, tamper resistance (e.g. coating PUFs) and unclonability (all PUFs). In addition, if the attacker was to tamper with the PUF, the mutual authentication step at the end of the protocol would fail since the Rep procedure would generate a different key from the one generated (and sent to the end-user) by the TTP. Regarding forward secrecy, there is no universal key stored in the base station from which node keys are derived. Every node has a different key and compromising any node's key does not give any information about a different node's key.

The verification step requires that the node M be able to generate a random nonce η_2. Low-power random number generators have been proposed in [28, 8]. Both approaches make use of pseudo-random number generators based on a keyed MAC algorithm and an incrementing counter. That poses the question of what key (call it K_{RNG}) to use. Since we are only using the key to generate a random number, one could simply use the deployed K_M for that purpose. If for some reason, the key had been tampered with, then the verification protocol will fail on the user/base-station side and the user can take appropriate measures. An alternative is to use $K_{RNG} \leftarrow \mathsf{Hash}(K_M||i)$, where i is a random bit and Hash is a collision resistant hash function. The random bit could originate from the PUF response which is only available within the sensor node. This would prevent the attacker from choosing a key that creates a known nonce η_2, since the attacker is then not able to predict the value of K_{RNG}. Another possibility is to use a random number generator based on PUFs as described in [26]. Finally, we notice that it is also possible to achieve the verification without generating a random nonce in the node. This can be achieved by adding a second round of communication in which the TTP sends the value $h_M \leftarrow \mathsf{Hash}(K_M||W_M)$ to the HWM and the HWM stores it in the node M. No knowledge of the key K_M is disclosed thanks to the properties of the hash function. An attacker could tamper with the value h_M and with W_M during the verification phase of the protocol. However, the verification step will nevertheless fail since it depends on knowledge of the key K_M and of the hash value h_M, neither of which the attacker can compute thanks to the properties of hash functions and of secure fuzzy extractors (i.e. you obtain negligible information about K_M from W_M).

3.4 Related Work on Secure Key Deployment for Sensor Nodes

Though there has been a lot of work on different key deployment schemes for sensor networks (such as ZigBee [35], SPINS [28], LEAP [34], Transitory Master Key [11], and random key pre-distributions [10, 13, 14, 24, 29]), most of them assume the initial secret key to be on the sensor node based on an unspecified security mechanism. However, there are also other key establishment procedures which address the initial key exchange like the Message-In-a-Bottle [21], Resurrecting Duckling [31], Talking to Strangers [2], Seeing-is-Believing [25], On-off Keying [6], Key Infection [1], and Shake Them Up [9]. These protocols differ on various security considerations, ease of use and associated costs.

Using an out-of-band channel physical contact is the method that is used in the Resurrecting Duckling to securely share a secret key [31]. This method can securely and authentically share a secret-key between devices if the direct contact channel is assumed to be secure. It also gives demonstrative identification and is robust to user errors. However, the main disadvantage is the need for extra hardware per-node to enable the information exchange through a physical contact. Another out-of-band channel based method is Talking-to-Strangers which uses a location-limited channel like infrared or audio to setup a public

key [2]. This method is similarly not cost effective both due to the need for extra specialized hardware per sensor node for the communication and the use of public-key cryptography. Seeing-is-Believing methods use a public-key encoded as a 2D-barcode to set up the key [25]. Unlike the Talking-to-Strangers protocol, Seeing-is-Believing requires only a single specialized set-up hardware equipped with a camera or barcode reader. However, it does require costly public-key cryptography to be performed on the sensor nodes. The Shake-Them-Up scheme sets up keys among nodes by holding a node in each hand and shaking them. The nodes exchange identical packets and thus, the attacker is not able to distinguish between messages originating from either device and transmitted on the same wireless channel [9]. To avoid the attacker spatially distinguishing the sources based on the power, the devices are shaken together during this communication. This approach, however, is not fully secure due to radio fingerprinting [30]. Though it provides physical identification of the devices, the key could be compromised if the user does not shake sufficiently. Smart-Its Friends [19] and Are-You-with-Me [22] are related schemes but requiring additional accelerometer on the nodes to measure movement. The On-off Keying technique uses the presence or absence of the RF signal to encode a 1 or a 0 respectively [6]. Assuming the attacker could only modify a 0 (RF absence) to 1 with an RF signal, then the message can still be authenticated by encoding it appropriately. Authenticity cannot be completely guaranteed as the authors of the scheme do not specify how the devices know what the authentic levels of 1 and 0 are. The scheme also requires the use of public-key cryptography and lacks a physically demonstrative identification of the devices with which keys are shared. Key Infection is just a simple and cost effective scheme assuming that the attacker is not present at the moment the keys are shared [1]. Hence, the keys are sent in the clear which breaks both the security and authenticity because the key exchange could be performed also by an adversary. Clearly, such a scheme contradicts our security model in which it is assumed that the attacker is present before, during and after the key set-up procedure.

3.5 Comparison

Kuo et al. [21] provided an extensive comparison of their protocol with previous ones in their work. We augment their table with our two new protocols. We also add to the table the category tamper evidence and unclonability as shown here as Table 1. One can argue that our solution with a TTP does not provide

	This paper (with TTP)	This paper (without TTP)	Message-In-a-Bottle [21]	Resurrecting Duckling [31]	Talking to Strangers [2]	Seeing-is-Believing [25]	On-off Keying [6]	Key Infection [1]	Shake Them Up [9]
Security									
Key secrecy	Y*	Y	Y	Y	–	–	–	N	N
Key authenticity	Y	Y	Y	Y	Y	Y	N	N	Y
Key unclonability and tamper evidence	Y	Y	N	N	N	N	N	N	N
Usability									
Demonstrative identification	Y	Y	Y	Y	Y	Y	N	N	Y
Robust to user error	Y	Y	Y	Y	Y	Y	Y	Y	N
Costs									
No per-node extra hardware	Y	Y	Y	N	N	Y	Y	Y	Y
No specialized set-up hardware	Y	N	N	Y	Y	N	Y	Y	Y
No public-key cryptography	Y	Y	Y	Y	N	N	N	Y	N

Table 1. Comparison of different key deployment techniques based on [21]. A '–' signifies hat this property is not applicable. Y* signifies that it provides this property is a slightly different adversarial model.

key secrecy in the same sense that MIB or our solution without TTP. However, it would also not be adequate to say that it offers no key secrecy since the only eavesdroppers that can compromise the key are the TTP and the hardware manufacturer which, depending on the application, are trusted. In addition, for the key to be compromised by the HWM, the HWM has to be active, i.e., it should be actively trying to eavesdrop the communications of the user and be present during key deployment. This is in sharp contrast with other protocols [1, 9] in which *any* eavesdropper can compromise the secrecy of the key. In addition, PUFs provide another type of security guarantee implied by their unclonability and tamper

evidence. Such property is only available to PUF-based solutions. PUFs also provide simplifications in the protocols. This is particularly true if we look at the number of rounds of communication in our newly proposed protocols and compare this number to those of the MIB protocol. In the case of the TTP-based protocol, PUFs also allow to get away without any specialized set-up hardware, which will certainly reduce costs. It is also important to point out the advantages that a PUF-based solution has over a solution based on burning the key in the node's ROM memory. Such a ROM-based solution allows the HWM to know the key without any effort and provides no guarantees as to whether the key has been tampered with by the time the end-user gets the sensor node.

4 Conclusions

The promise of ambient intelligence will only achieve its true potential if we can guarantee that the information gathered around us is used in a privacy sensitive and secure manner. This, in turn, can only be achieved if we trust that the keys used to secure our sensitive information have not been compromised. In this paper, we have described how PUFs and their corresponding Helper Data algorithm (or Fuzzy Extractor) can help us achieve these goals. In particular, we introduce two protocols for secure key deployment in the absence of any (previously) shared secret. Our protocols take advantage of specific fuzzy extractor properties to provide secrecy and authenticity of the deployed key against omni-present adversaries, i.e., adversaries that are present everywhere and all the time. Compared to previous protocols, and most prominently the Message-In-a-Bottle proposal [21], our protocols are simpler (less communication complexity) and require less additional hardware. In addition, because of the use of PUFs, our solution provides tamper evidence and unclonability.

References

1. R. Anderson, H. Chan, and A. Perrig. Key Infection: Smart Trust for Smart Dust. In *IEEE International Conference on Network Protocols — ICNP 2004*, pages 206–215. IEEE Computer Society, October 5-8, 2004.
2. D. Balfanz, D. K. Smetters, P. Stewart, and H. Chi Wong. Talking to Strangers: Authentication in Ad-Hoc Wireless Networks. In *Network and Distributed System Security Symposium — NDSS 2002*, 2002.
3. M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In D. R. Stinson, editor, *Advances in Cryptology — CRYPTO '93*, volume 773 of *LNCS*, pages 232–249. Springer, August 22-26, 1993.
4. N. Bird, C. Conrado, J. Guajardo, S. Maubach, G.-J. Schrijen, B. Škorić, A. M. H. Tombeur, P. Thueringer, and P. Tuyls. ALGSICS - Combining Physics and Cryptography to Enhance Security and Privacy in RFID Systems. In F. Stajano, C. Meadows, S. Capkun, and T. Moore, editors, *Security and Privacy in Ad-hoc and Sensor Networks — ESAS 2007*, volume 4572 of *LNCS*, pages 187–202. Springer, July 2-3, 2007.
5. X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. Smith. Secure remote authentication using biometric data. In R. Cramer, editor, *Advances in Cryptology — Eurocrypt 2005*, volume 3494 of *LNCS*, pages 147–163. Springer-Verlag, 2005.
6. M. Cagalj, S. Capkun, and J. Hubaux. Key agreement in peer-to-peer wireless networks. *Proceedings of the IEEE (Special Issue on Cryptography and Security)*, 94(2), 2006.
7. L. Carter and M. N. Wegman. Universal Classes of Hash Functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979.
8. C. Castelluccia and A. Francillon. TinyRNG, A Cryptographic Random Number Generator for Wireless Sensor Network Nodes. In *International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks — IEEE WiOpt 2007*. IEEE, April, 2007.
9. C. Castelluccia and P. Mutaf. Shake them up!: a movement-based pairing protocol for CPU-constrained devices. In K. G. Shin, D. Kotz, and B. D. Noble, editors, *International Conference on Mobile Systems, Applications, and Services — MobiSys '05*, pages 51–64. ACM, 2005.
10. H. Chan, A. Perrig, and D. Song. Random Key Predistribution Schemes for Sensor Networks. In *IEEE Symposium on Security and Privacy — S&P 2003*, pages 197–215. IEEE Computer Society, 2003.
11. J. Deng, C. Hartung, R. Han, and S. Mishra. A practical study of transitory master key establishment forwireless sensor networks. In *International Conference on Security and Privacy for Emerging Areas in Communications Networks — SECURECOMM'05*, pages 289–302, Washington, DC, USA, 2005. IEEE Computer Society.
12. Y. Dodis, M. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology —- EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 523–540. Springer-Verlag, 2004.

13. W. Du, J. Deng, Y. S. Han, and P. K. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. In S. Jajodia, V. Atluri, and T. Jaeger, editors, *ACM Conference on Computer and Communications Security — CCS 2003*, pages 42–51, New York, NY, USA, 2003. ACM.
14. L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In V. Atluri, editor, *ACM Conference on Computer and Communications Security — CCS 2002*, pages 41–47, New York, NY, USA, 2002. ACM.
15. B. Gassend, D. E. Clarke, M. van Dijk, and S. Devadas. Silicon physical unknown functions. In V. Atluri, editor, *ACM Conference on Computer and Communications Security — CCS 2002*, pages 148–160. ACM, November 2002.
16. J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls. FPGA Intrinsic PUFs and Their Use for IP Protection. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems — CHES 2007*, volume 4727 of *LNCS*, pages 63–80. Springer, September 10-13, 2007.
17. J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls. Physical Unclonable Functions and Public Key Crypto for FPGA IP Protection. In *International Conference on Field Programmable Logic and Applications — FPL 2007*, pages 189–195. IEEE, August 27-30, 2007.
18. D. E. Holcomb, W. P. Burleson, and K. Fu. Initial SRAM state as a fingerprint and source of true random numbers for RFID tags. Conference on RFID Security 07, July 11-13, 2007.
19. L. E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl, and H.-W. Gellersen. Smart-its friends: A technique for users to easily establish connections between smart artefacts. In *Ubicomp 2001: Ubiquitous Computing, Third International Conference*, pages 116–122, 2001.
20. A. Juels and M. Wattenberg. A Fuzzy Commitment Scheme. In J. Motiwalla and G. Tsudik, editors, *ACM Conference on Computer and Communications Security — ACM CCS '99*, pages 28–36. ACM, November 1-4, 1999.
21. C. Kuo, M. Luk, R. Negi, and A. Perrig. Message-In-a-Bottle: User-Friendly and Secure Key Deployment for Sensor Nodes. In *International Conference on Embedded Networked Sensor Systems — SenSys '07*, pages 233–246. ACM, 2007.
22. J. Lester, B. Hannaford, and G. Borriello. "are you with me?" - using accelerometers to determine if two devices are carried by the same person. In *Pervasive Computing, Second International Conference*, pages 33–50, 2004.
23. J.-P. M. G. Linnartz and P. Tuyls. New Shielding Functions to Enhance Privacy and Prevent Misuse of Biometric Templates. In J. Kittler and M. S. Nixon, editors, *Audio-and Video-Based Biometrie Person Authentication — AVBPA 2003*, volume 2688 of *LNCS*, pages 393–402. Springer, June 9-11, 2003.
24. D. Liu, P. Ning, and W. Du. Group-based key pre-distribution in wireless sensor networks. In M. Jakobsson and R. Poovendran, editors, *ACM Workshop on Wireless Security — WiSe 2005*, pages 11–20, New York, NY, USA, 2005. ACM.
25. J. M. McCune, A. Perrig, and M. K. Reiter. Seeing-Is-Believing: Using Camera Phones for Human-Verifiable Authentication. In *IEEE Symposium on Security and Privacy — S&P 2005*, pages 110–124. IEEE Computer Society, May 8-11, 2005.
26. C.W. O'Donnel, G.E. Suh, and S. Devadas. PUF-Based Random Number Generation. Technical Memo MIT-CSAIL-CSG-481, MIT CSAIL, November 2004.
27. R. S. Pappu. *Physical one-way functions*. PhD thesis, Massachusetts Institute of Technology, March 2001. Available at `http://pubs.media.mit.edu/pubs/papers/01.03.pappuphd.powf.pdf`.
28. A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler. SPINS: Security Protocols for Sensor Networks. *Wireless Networks*, 8(5):521–534, 2002.
29. M. Ramkumar and N. Memon. An efficient key predistribution scheme for ad hoc network security. *IEEE Journal on Selected Areas in Communications*, 23(3):611–621, 2005.
30. K. B. Rasmussen and S. Capkun. Implications of radio fingerprinting on the security of sensor networks. In *International Conference on Security and Privacy in Communication Networkds — SecureComm 2007*. IEEE, September 17-20, 2007.
31. F. Stajano and R. J. Anderson. The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. In B. Christianson, B. Crispo, J. A. Malcolm, and M. Roe, editors, *Security Protocols*, volume 1796 of *LNCS*, pages 172–182. Springer-Verlag, April 19-21, 1999.
32. P. Tuyls, G.-J. Schrijen, B. Škorić, J. van Geloven, N. Verhaegh, and R. Wolters. Read-Proof Hardware from Protective Coatings. In L. Goubin and M. Matsui, editors, *Cryptographic Hardware and Embedded Systems — CHES 2006*, volume 4249 of *LNCS*, pages 369–383. Springer, October 10-13, 2006.
33. B. Škorić, T. Bel, A.H.M. Blom, B.R. de Jong, H. Kretschman, and A.J.M. Nellissen. Randomized resonators as uniquely identifiable anti-counterfeiting tags. Technical report, Philips Research Laboratories, January 28th, 2008.
34. S. Zhu, S. Setia, and S. Jajodia. Leap+: Efficient security mechanisms for large-scale distributed sensor networks. *ACM Transactions on Sensor Networks*, 2(4):500–528, 2006.
35. ZigBee Specification. Technical Report Document 053474r06, Version 1.0, ZigBee Alliance, June 2005.

Appendix

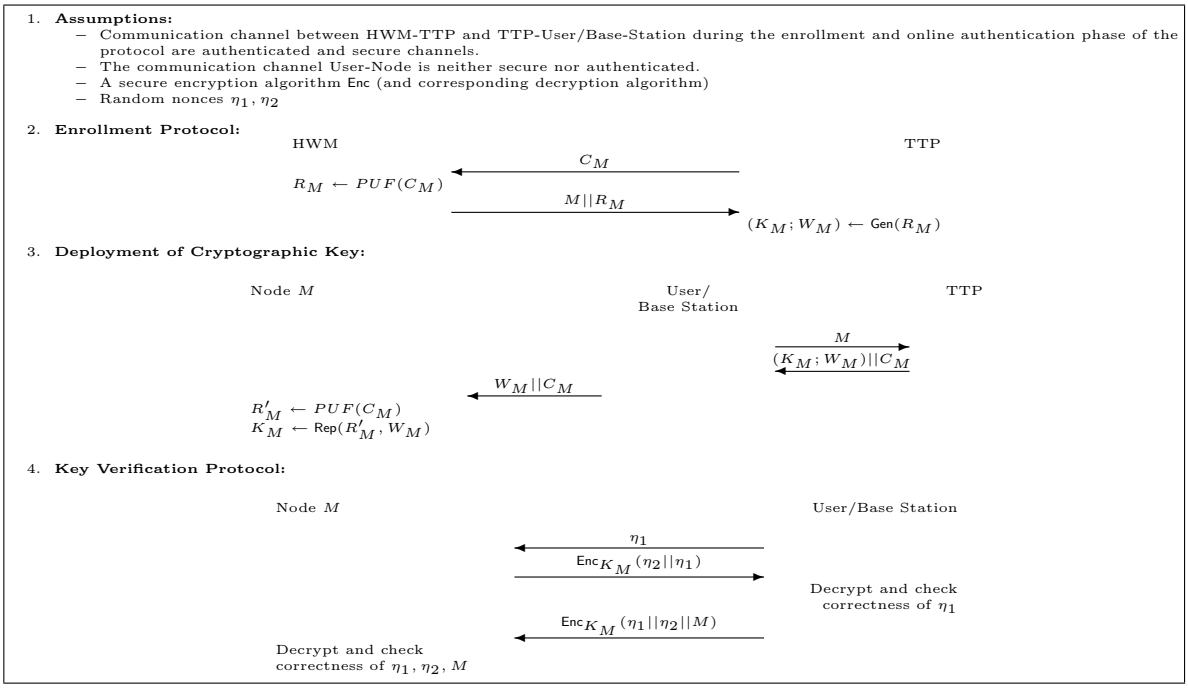
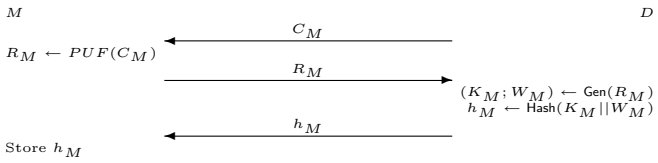


Fig. 1. Key deployment protocol with TTP

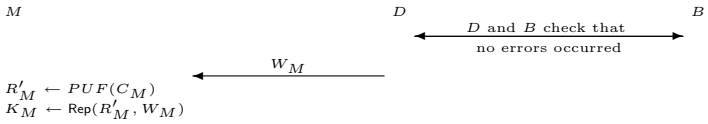
1. **Assumptions:**
   - Communication channel D–S and B–S are authenticated and secure channels during the set-up phase of the protocol.
   - The communication channel User-Node is neither secure nor authenticated.
   - A secure encryption algorithm Enc (and corresponding decryption algorithm)
   - A random nonce η
   - PUF as described in Section 2
2. **Set-up:**

3. **Placement in Faraday Cage:** Node M and keying device are placed in the Faraday cage and the Faraday cage is closed. The keying beacon is placed outside the Faraday cage to jam all communications in the frequency used by the keying device and the node M.

4. **Deployment of Cryptographic Key (inside Faraday Cage):**



5. **Key Activation and Protocol Check (outside Faraday Cage):**



6. **Key Verification Protocol:**

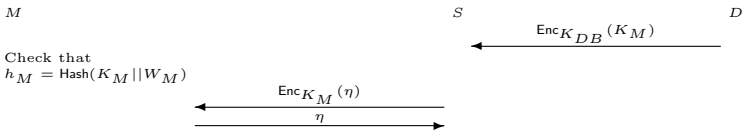


Fig. 2. Key deployment protocol without TTP