Secure Initialization of Multiple Constrained Wireless Devices for an Unaided User

Toni Perković, *Member, IEEE,* Mario Čagalj, *Member, IEEE,* Toni Mastelić, Nitesh Saxena, *Member, IEEE,* Dinko Begušić, *Member, IEEE,*

Abstract—A number of protocols and mechanisms have been proposed to address the problem of initial secure key deployment in wireless networks. Most existing approaches work either with a small number of wireless devices (i.e., two) or otherwise rely on the presence of an auxiliary device (such as a programmable camera, computer or Faraday cage). In this paper, we design a solution that allows a user unaided initialization (free from auxiliary devices) of a relatively large number of wireless devices. The proposed solution is based on a novel multichannel *Group message Authentication Protocol* (GAP), in which information is transmitted over both a radio and a visible light channel (VLC). A notable feature of GAP is that the information to be authenticated is independent of the *short authentication string* to be verified by the user (an *indirect* binding protocol [28]). This, as we show, results in a lower communication cost compared to existing *direct* binding protocols. The advantage in terms of the communication cost of our GAP protocol is especially important for power-constrained devices, such as wireless sensor motes.

Another appealing feature of GAP is that it is secure in the attacker model where the VLC is *semi-authentic*; whereas existing protocols consider VLC to be authentic. This is made possible by using joint Manchester-Berger unidirectional error-detection codes that are secure and easy to interpret by a non-specialist and unaided end user. Our overall key deployment mechanism has minimal hardware requirements: one LED, one button and, of course, a radio transceiver, and is thus suitable for initializing devices with *constrained* interfaces, such as (multiple) wireless sensor motes. We demonstrate the feasibility of the proposed method via a preliminary usability study. The study indicates that the method has reasonably low execution time, minimal error rate and is user-friendly.

Index Terms—Message authentication protocol, Out-of-Band Communication, Usable security, Wireless networks

1 Introduction

Wireless Sensor Networks (WSN) are increasingly gaining momentum in our lives. Tomorrow's e-healthcare systems, smart homes, power management systems will involve a large number of inter-connected smart wireless (sensor) devices that will be operated and controlled by end users (a home user or an administrator). These devices have the capability to connect and interact, and provide a backbone for the future development of the "Internet of Things". In a WSN environment, the nodes might need to communicate security sensitive data among themselves and with the base station (also referred to as "sink"). The communication among the nodes might be point-to-point and/or broadcast, depending upon the application. These communication channels, however, are easy to eavesdrop on and are easy to manipulate, raising the very real threat of the so-called *man-in-the-middle* attacker. A fundamental task, therefore, is to secure these communication channels.

1.1 Motivation for Secure Initialization

A number of so-called "key pre-distribution" techniques to bootstrap secure communication in a WSN have been proposed, e.g., [31], [13], [24], [12], [10]. However, all of these techniques assume that, before deployment, sensor nodes are somehow pre-installed with secret(s) shared with other sensor nodes and/or the sink. The TinySec architecture [15] also assumes that the nodes are loaded with shared keys prior to deployment. This might be a reasonable assumption in some, but certainly not all, cases. Let us consider, for example, a user-centric application of WSN. An individual user (Bob) wants to install a sensor network to monitor the perimeter of his property; he purchases a set of commodity noise and vibration sensor nodes at certain retailers, and wants to deploy the sensor nodes with his home computer acting as the sink. Being off-the-shelf, these sensor nodes are not sold with any built-in secrets. Some types of sensor nodes might have a USB (or similar) connector that allows Bob to plug each sensor node into his computer to perform secure initialization. This would be immune to both eavesdropping and man-in-the-middle attacks. However, most sensor nodes might not have any wired interfaces, since having a special "initialization" interface influences the complexity and the cost of the sensor node. Also, note that Bob would have to perform security initialization manually and separately for each sensor node. This undermines the scalability of the approach since potentially a reasonably large number of sensor nodes might be involved.

Furthermore, keys can not always be pre-loaded during the manufacturing phase because eventual customers might not trust the manufacturer, for example

- *T. Perković, M. Čagalj, T. Mastelić and D. Begušić are affiliated with the Department of Electrical Engineering, FESB, University of Split, Croatia. E-mail: {toperkov, mcagalj, toni.mastelic, begusic}@fesb.hr,*
- *N. Saxena is affiliated with the Department of Computer Science and Engineering, Polytechnic Institute of New York University, USA. E-mail: nsaxena@poly.edu.*

in WSNs deployed for military applications. Moreover, a WSN application might involve nodes produced by multiple manufacturers. Due to this reason, establishing pre-shared secrets or a PKI-based solution might be infeasible as it would require a global infrastructure involving many diverse manufacturers. We note that the problem of secure WSN initialization that we consider in this paper is very similar to the well-studied problem of "wireless (two-device) pairing", the premise of which is also based on the fact that the devices wanting to communicate with each other do not share any pre-shared secrets or a common PKI with each other [4], [16].

1.2 Requirements for Secure Initialization

An important research challenge, in light of the above discussion, is to design secure WSN initialization mechanisms that satisfy the following properties:

1) *User-friendliness*: The mechanism should be easily administered by a non-specialist and unaided end user. By unaided, we mean that the user is not in possession of any auxiliary devices that can facilitate or automate the initialization process. It is important to note that auxiliary devices may not always be available. They will also add to the overall cost of the system. Furthermore, requiring a specialized auxiliary device only for the sake of a security operation presents deployment hurdles.
2) *Scalability:* The mechanism should be able to initialize a reasonably large number of nodes. Due to the manual nature of initialization (because of the lack of an auxiliary device as stated above), however, one can only hope to initialize a maximum of, for example, 20-30 devices per batch of initialization. (The mechanism can be repeated in multiple batches whenever needed).
3) *Compatibility with Constrained Resources:* Being mass produced, sensor devices are usually constrained, i.e., they do not usually have wired or other traditional interfaces, such as displays and keypads. Thus, the initialization mechanism should be able to work within these resource constraints. In other words, secure initialization should still be possible even with a few on-board LEDs and buttons[1]. Because sensor-motes are typically on a limited power supply, an additional goal is to minimize the communication overhead associated with the secure initialization mechanism.

In addition to the necessary requirement of providing security against the man-in-the-middle attacker, it is desirable that the initialization provides protection against compromised nodes. This is needed to address scenarios such as those whereby a manufacturer sneaks in malicious sensor node(s) along with normal sensor nodes shipped to a customer, as pointed out in [17].

1. Most commercially available sensor motes and devices possess multiple LEDs and an on/off button (Mica2 [2])

1.3 Prior Work

The problem of secure initialization of sensor devices has received considerable attention by the research community and a number of solutions have been proposed. The prior solutions do not satisfy one or more of the requirements outlined above, however. Many existing solutions work only with a small number of (i.e., two) wireless devices and are not scalable. These include the "Shake-them-up" [9] scheme that suggests a simple manual technique for pairing two sensors that involves shaking and twirling them in very close proximity to each other, in order to prevent eavesdropping. Another scheme "Are You with Me" [21] uses human-controlled movement to establish a secret key between two devices.

The other notable recent result "Message-in-a-Bottle" [17] explores the use of a *Faraday Cage* to shield communication from eavesdropping and outside interference and allow a set of sensors to be simultaneously paired with the sink. This is a scalable technique, although as illustrated in [17], building a truly secure Faraday Cage is a challenge. The primary issue with this approach is the need to obtain and carry around a specialized piece of equipment – a Faraday Cage (an auxiliary device). The cost and the physical bulk of the cage can be problematic in practice.

Another well-established approach to securing initial key deployment involves two communication channels – an insecure high bandwidth radio channel and a low-bandwidth Out-of-Band (OoB) channel, such as visible light. The security of this approach relies on the assumption that the underlying OoB channel, being human-perceptible, is authenticated and preserves the integrity of transmitted messages. This approach has also been discussed in the 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) RFC4919 specification [1]. According to this specification, one of the major security considerations involves application of "out-of-band techniques for the initial key establishment" among a large number of sensor devices.

Many prior solutions based on the above multi-channel approach, however, rely on the presence of auxiliary devices. For example, the solutions presented in [36], [37], [35], [30], [22], [41] all require a programmable video camera. Yet other solutions (e.g., GAnGs [11] and Groupthink [29]) are geared for multi-user group settings whereby each user is in possession of a personal device, such as a smartphone. These solutions require interfaces beyond the reach of current sensor devices, such as full displays or cameras.

1.4 Our Contributions

In this paper, we present the first secure and usable initialization mechanism that works with multiple (sensor) devices having constrained resources (a LED, a button and limited power supply) and does not require any auxiliary devices, thus satisfying all the requirements outlined previously.

Our initialization mechanism is based on a novel (multichannel) protocol, called the *Group message Authentication Protocol* - GAP. GAP involves communication over a radio channel and an out-of-band visible light channel (VLC). GAP is inspired by the two-party SAS protocol [40], [8]; we show that straightforward generalizations of SAS to a multiparty protocol may easily fall short of being secure. A notable feature of GAP is that the information to be authenticated is independent of the *short authentication string* (an *indirect* binding protocol [28]) to be verified by the user over a visible light channel (i.e. the GAS and authenticated information are completely independent in the sense of probability). This, as we show, results in a lower communication cost compared to existing *direct* binding protocols. The advantage in the communication cost of our GAP protocol is especially important for battery-powered devices, such as wireless sensor nodes. We also show how to secure GAP against malicious insider attacks (compromised sensor nodes); in [22], the devices are assumed to be benign during the initialization phase.

As we show later, the visible light channel (VLC) is prone to certain bit-manipulation attacks, that (contrary to the common belief) renders VLC *semi-authenticated*; however many existing protocols [36], [37], [35], [22] that use VLC consider it to be authenticated. In order to prevent these attacks, we use a simple combination of well known unidirectional codes (Berger and Manchester), which is easy to interpret by an end user.

Finally, we demonstrate the feasibility of the proposed mechanism via a preliminary usability study with 28 users. The study indicates that the method has reasonably low execution time, minimal error rate and is user-friendly. We further discuss how the usability and scalability of our mechanism can be improved by utilizing a zero-configuration auxiliary device (e.g., a standard camera phone with no additional computational logic), when available.

We note that although we target our scheme as a means of secure initialization of a WSN, our proposal is also equally applicable to other wireless devices scenarios. This includes, for example, the initialization of a number of commodity wireless access points that need to be installed as part of an enterprise's wireless network.

Paper Outline. In Section 2, we state the assumptions and a give an overview of our solution. In Section 3, we present the GAP protocol, and in Section 4, we give a solution against insider attacks. Section 5 deals with securing of transmissions over a visible light channel. We discuss several implementation aspects of GAP in Section 6. Usability evaluation is presented in Section 7. Related work is provided in Section 8, and we conclude in Section 9.

2 Assumptions and Solution Overview

In this section, we outline our attacker model and assumptions, and provide an overview of the secure initialization protocol and mechanism.

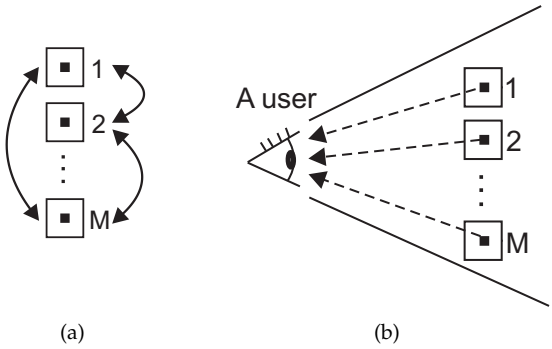


Fig. 1. Two phases of GAP: in (a) devices exchange messages to be authenticated over a radio channel and (b) a user performs authentication via a visible light channel (dashed arrow).

2.1 Attacker Model

In our initialization protocol the information is sent over two channels: a radio and a visible light channel. We assume that an attacker has a full control over the radio channel; he can eavesdrop, drop, delay, replay and modify messages sent on this channel. He can initiate communication with any device and at any given time. The attacker can also eavesdrop and modify messages sent over a visible light channel (VLC) at all times (Section 5); the attacker however cannot disable the visible light communication channel (erase the messages). To convey information via VLC we use on-off keying (i.e., bit "0": LED OFF, bit "1" LED ON). Note that an attacker equipped with a directional light source (e.g. a laser) can potentially modify bits sent via VLC. With such keying 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 power OFF. In this case, in our model, we speak of a *semi-authenticated* visible light channel. For this reason we apply error detection codes to the group authentication string before its transmission over VLC; in Section 5 we show that such coding prevents the bit flipping attacks. Note that the existing related approaches [36], [37], [35], [22], [41] consider the visible channel to be authenticated (i.e., the attacker does not control messages sent over VLC). These protocols are therefore insecure in our model and we work in a much stronger attacker model. To start with, we assume that devices involved in key deployment are not compromised. Later in Section 4, we extend this attacker model to include stronger insider attacks from compromised wireless devices.

2.2 Solution Overview

A user wishes to initialize a larger set of wireless sensor devices. She makes sure that the sensor devices are all placed in her visual field, so that she can simultaneously observe their LEDs (Figure 1(b)). She powers on the devices, picks an arbitrary one and designates it as a *coordinator* node. Once this has been done, the

coordinator initiates the execution of our *Group message Authentication Protocol* (GAP) that enables mutual authentication of messages (e.g. public keys) exchanged by the devices over an insecure radio channel (Figure 1(a)). Our protocol accomplishes this by first generating a common short *group authentication string* (GAS) on all the devices. In turn, the GAS is communicated over a semi-authenticated visible light channel. Finally, the user visually verifies that the GAS transmitted over the VLC is the same on all the devices (Figure 1(b)). If the verification is successful the user completes the protocol by pushing the button on each device.

A notable feature of our *Group message Authentication Protocol* is that it binds authenticated information (e.g. public keys) *indirectly* to a group authentication string [28]. In indirect binding protocols, the GAS is functionally independent of the information to be authenticated (i.e. the GAS and authenticated information are completely independent in the sense of probability [28]). While there is no relation between the compared GAS and the authentic information, the security of the protocols comes from some mechanism that binds short random nonces (used for calculation of the GAS) and the authentic information together in a secure way. In our case, this secure binding is achieved through the commitment scheme. For this reason in GAP the devices exchange messages to be authenticated (e.g., public keys) through specially formed commit/decommit pairs over a radio channel (Section 3 and Figure 2).

On the other hand, the direct binding approach requires the GAS to be dependent on the information devices want to authenticate. The basic principle behind the direct binding approach is to make all the parties, who are intended to be part of a protocol run, agree on a short-output hash or digest of a complete description (the collection of all the information that any member of the group wishes to have authenticated to) of the protocol run [28].

We show in Section 3.3 that indirect binding, as used in our GAP protocol, can reduce the communication cost compared to the representative directly binding protocol proposed by Laur and Pasini in [18], [19].

3 Efficient and Secure Group Message Authentication Protocol

In this section we provide details of our *Group message Authentication Protocol* - GAP. The GAP protocol shares some similarities with the existing group authentication protocols (please see an excellent survey on such protocols by Nguyen and Roscoe [28]). Still, it is different in two important ways:

1) The GAP binds authenticated information (e.g. public keys) *indirectly* to a *group authentication string* [28]. As a result, GAP incurs *lower communication cost* compared to some (provably secure) directly binding protocols (Table 1). This is especially important for battery-powered devices such as wireless sensor devices. It appears that this disadvantage is common to all (provably secure) directly binding schemes that use universal hash functions to generate short group authentication string (as discussed later in the section).
2) The GAP is a *secure* generalization of the two party SAS protocol [8] into a multiparty version. This is in contrast with recent Nguyen and Roscoe's attempt in [28] to generalize the SAS protocol. Indeed, we show in Section 3.2 that their straightforward generalization results in an insecure protocol. We state the security result for GAP in Theorem 1.

Two aspects of the GAP protocol are essential for its security: (i) GAP imposes strict ordering among the messages exchanged by the devices and (ii) at least one device from the group (e.g. a coordinator) must know the correct group size. Please note that the group size information is entered by the user. This process can be very challenging with devices that have constrained interfaces (e.g. a single pushbutton and one LED). We describe a possible user-friendly approach for accomplishing this task in Section 6 and study its usability aspects in Section 7. We next give details of our GAP protocol.

3.1 Description of the GAP

Let us introduce some notation. A user wishes to initialize a set of M sensor devices. We assume that sensor devices involved in key deployment are trusted. We denote this group with $\mathcal{G}$, i.e., $\mathcal{G} = \{ID_1, ID_2, \ldots, ID_M\}$, ID_j being the identity of the jth device (IDs are unique). The set $\mathcal{G}$ is ordered with respect to the increasing identities ($ID_1 < ID_2 < \ldots ID_M$). Let PK_j, $j \in \mathcal{G}$, denote a public key of wireless device j (i.e., the device with the identifier ID_j). For clarity, variables $\widehat{c}_j$, $\widehat{d}_j$, $\widehat{N}_j$ and $\widehat{h}_{\mathcal{G}_j}$ denote the variables from device with identifier ID_j (or shortly device j) received by the device i (i.e., the device with the identifier ID_i). The hats in the notation indicate a possible modification (or influence) by an adversary. We denote with k the *coordinator* sensor device. The GAP protocol evolves as shown in Figure 2. The goal of the protocol is to assist the devices in $\mathcal{G}$ to mutually authenticate their respective public keys.

Phase I (radio). The user first designates one arbitrary sensor device as a coordinator; this is achieved by a push on a button (Section 5). We denote this device with $k \in \mathcal{G}$. Upon selecting the coordinator $k \in \mathcal{G}$, it starts to broadcast its ID_k to all other nodes. Upon receiving the ID_k from the coordinator, the other nodes from $\mathcal{G}$ begin to broadcast their own IDs until the predefined timeout (a couple of seconds). Each device $i \in \mathcal{G}$ orders all received IDs in the order of increased identities ($ID_1 < ID_2 < \ldots < ID_M$). We use $\mathcal{G}_i$ to denote the ordered set of IDs as seen by device $i \in \mathcal{G}$.

Phase II (radio). A commitment scheme is an important cryptographic building block that is used in our GAP. A commitment function transforms a value m into

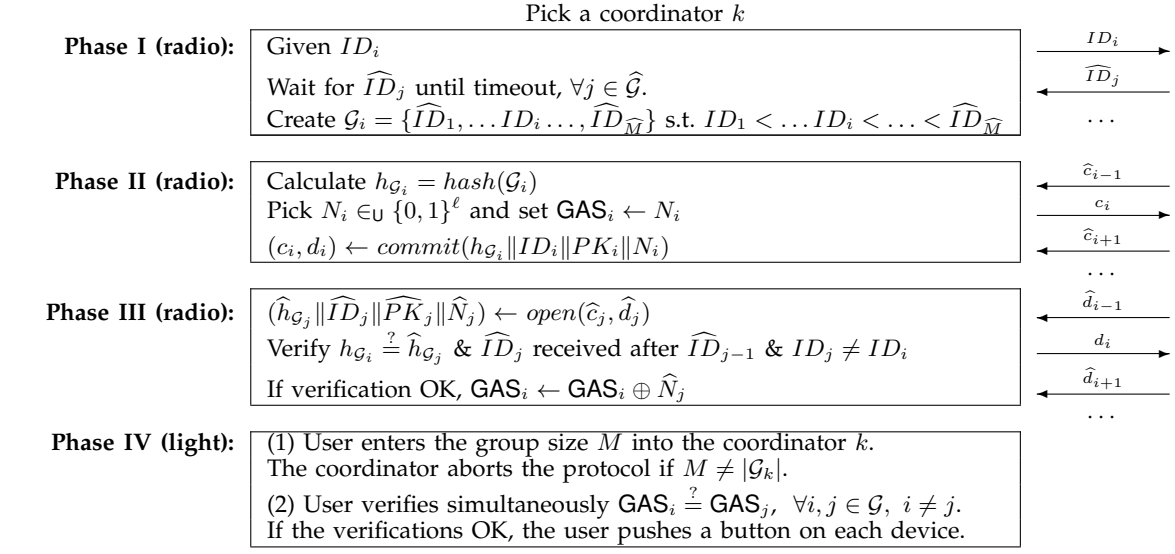


Fig. 2. Group message Authentication Protocol (GAP): Authenticating public keys PK_i ($i \in \mathcal{G}$) using a Group Authentication String (GAS).

a commitment/opening pair (c, d), where c reveals no information about m, but (c, d) together reveal m, and it is infeasible to find $\widehat{d}$ such that reveals $\widehat{m} \neq m$. Every device $i \in \mathcal{G}$ calculates a hash value $h_{\mathcal{G}_i}$ of the set $\mathcal{G}_i$, generates random nonce N_i (ℓ bits). Next, the device i sends its commitment c_i to all other participants $j \in \mathcal{G}_i$ but only after having received commitments from all the devices $j \in \mathcal{G}_i$, such that $ID_j < ID_i$. Note the each device $i \in \mathcal{G}$ adds the $h_{\mathcal{G}_i}$ and its ID_i to the commitment. This is used to prevent reflection and node injection attacks.

Phase III (radio). Decommitments are sent in the same order as the corresponding commitments. Upon receiving decommitment $\widehat{d}_j$ from $j \in \mathcal{G}$ the device i ($i \neq j$) opens the commitment $\widehat{c}_j$ and verifies the sender's ID and that $\mathcal{G}_i$ matches $\mathcal{G}_j$ (by comparing $h_{\mathcal{G}_i}$ and $\widehat{h}_{\mathcal{G}_j}$). If the verifications are OK, i updates GAS_i as follows: $\mathsf{GAS}_i \leftarrow \mathsf{GAS}_i \oplus \widehat{N}_j$. Otherwise, device i aborts the protocol. This process repeats for all received decommitments.

Phase IV (light). In the last phase of the protocol the user first enters the size of the group (M) into the coordinator that in turn verifies it to be equal to $|\mathcal{G}_k|$ (see Section 6 for details). Next, the user simultaneously verifies that the established GAS values on all the devices satisfy $\mathsf{GAS}_i = \mathsf{GAS}_j$ $\forall i, j \in \mathcal{G}$. In Section 5) we describe a secure method to accomplish this task, which is based on LED blinking. If the verification is OK, the user pushes a button on each device to complete the initialization process. At this stage, each sensor device from $\mathcal{G}$ holds authenticated public keys (or other messages) of all the other devices.

Next we state the security result for the GAP protocol. We assume that sensor devices involved in key deployment are trusted. Next, we assume the used hash function $hash(\cdot)$ to be collision resistant and the commitment scheme $commit(\cdot)/open(\cdot)$ to be non-malleable.

Theorem 1: The probability that a computationally bounded adversary breaks (in a single attempt) the GAP is bounded by $2^{-\ell} + \varepsilon$, where ℓ is the size (in bits) of the group authentication string (GAS) and ε is a negligible probability.

We provide the sketch of the proof of Theorem 1 in Appendix. The important implication of this result is that the authentication string GAS can be reasonably short (e.g., 15-20 bits). This is especially important given that the user verifies these bits without any assistance from auxiliary devices. In Appendix we proved the security of the GAP protocol in the attacker model in which the sensor nodes are trusted. However, GAP protocol is not secure against insider attacks (compromised nodes). Note that this is a realistic attack because an attacker, for example, can sneak in malicious sensor node(s) along with normal sensor nodes during the transportation phase. Later in Section 4.1 we propose a simple extension of the GAP protocol, which is secure against compromised insider devices, all this at a small additional communication cost.

As stated at the beginning of this section, the GAP protocol generalizes the two party SAS protocol into a multiparty version. Although it may appear at first that this is a straightforward task, we exemplify next that such belief may be unfounded.

3.2 Insecure SAS Protocol Generalization

In [28] Nguyen and Roscoe propose a straightforward generalization of the two party SAS protocol [8]. Similar to GAP, in their proposal group members in the first phase exchange commitment messages and subsequently, in the second phase, exchange the corresponding decommitments. However, Nguyen and Roscoe allow for arbitrary interleaving of the commitment and de-

TABLE 1
Comparison of GAP protocol and GMA [18] in terms of communication and computation.

		GMA [bits]	GAP [bits]	Difference: GMA−GAP [bits]
Tx/Rx cost	ID	$M \cdot \lvert ID\rvert$	$M \cdot \lvert ID\rvert$	0
	c	$M \cdot \lvert K_L\rvert/2$	$M \cdot \lvert K_L\rvert/2$	0
	d	$M \cdot (\lvert ID\rvert + \lvert K_L\rvert)$	$M \cdot (\lvert ID\rvert + \lvert K_L\rvert/2 + \lvert PK\rvert + \lvert N\rvert)$	$M \cdot (\lvert K_L\rvert/2 - \lvert PK\rvert - \lvert N\rvert)$
	PK	$M \cdot \lvert PK\rvert$	0 (part of the commit/open pair)	$M \cdot \lvert PK\rvert$
				$\mathbf{M \cdot (\lvert K_L\rvert/2 - \lvert N\rvert)}$
Comput. cost	hash $h(\cdot)$	$M \cdot \lvert K_L\rvert/2 \cdot (\lvert ID\rvert + \lvert PK\rvert)$	$M \cdot \lvert K_L\rvert/2 \cdot \lvert ID\rvert$	$M \cdot \lvert K_L\rvert/2 \cdot \lvert PK\rvert$
	$commit(\cdot)$	$M \cdot \lvert K_L\rvert/2 \cdot (\lvert ID\rvert + \lvert K_L\rvert)$	$M \cdot \lvert K_L\rvert/2 \cdot (\lvert ID\rvert + \lvert K_L\rvert/2 + \lvert PK\rvert + \lvert N\rvert)$	$M \cdot \lvert K_L\rvert/2 \cdot (\lvert K_L\rvert/2 - \lvert PK\rvert - \lvert N\rvert)$
	GAS	0	0	0
		$\lvert K_L\rvert = 2 \cdot length(h(\cdot))$	M - group size	$\mathbf{\lvert K_L\rvert \cdot M \cdot (\lvert K_L\rvert/2 - \lvert N\rvert)}$

commitment messages in their respective phases (which is not the case in our GAP). Such a construction (arbitrary interleaving) results in an insecure protocol, as we exemplify next. We show that an attacker can replace for example a public key (or any other message to be authenticated) of a legitimate device with the one of his own choosing. We will describe the attack in a two party scenario as a special case of a multiparty scenario. Let us assume that two devices ID_1 and ID_2 want to mutually authenticate their public keys PK_1 and PK_2, respectively. Following Nguyen and Roscoe's protocol [28] the devices send their respective commitments c_1 and c_2. Attacker A blocks commitment c_2 sent by ID_2 and waits to receive the corresponding decommitment d_2. Having received c_1 and transmitted c_2, the device ID_2 considers the first phase to be completed. ID_2 enters the second phase of the protocol and sends d_2 before receiving d_1; according to Nguyen and Roscoe this is a legitimate behavior [28]. Now the attacker A blocks d_2, opens the commitment c_2 and retrieves the random number N_2 from it. This allows A to create a commitment $\widehat{c_2}$ (in which he replaces PK_2 with his own $\widehat{PK_2}$) which he finally sends to ID_1. This in the end results in the same short group authentication string (i.e. $N_1 \oplus N_2$) at the devices ID_1 and ID_2, but A has succeeded in replacing ID_2's public key (PK_2) with the one of his own choosing ($\widehat{PK_2}$). While it is relatively easy to detect this flaw in the two party scenario, the problem arises in the multiparty setting where designers usually fail to realize that the ordering between exchanged messages has to be maintained between all the possible pairs of the group members. This is exactly what GAP does.

3.3 Communication cost: Indirect Binding vs. Direct Binding Protocols

Due to the potentially large number of sensor nodes and the requirement for a power source such as a battery, even small energy savings per device imply a significant "green potential" [34]. In this section we study potential advantages of directly over indirectly binding protocols in terms of communication cost. Nguyen and Roscoe in [28] give a simple model which compares the computation cost between various group pairing protocols. While Nguyen and Roscoe focused on the computational aspects of various group pairing protocols, we believe that in our setting it is more important to consider related communication cost, especially in low-power wireless sensor networks where the communication cost dominates the computation.

Recall from Section 2 in indirect binding protocols (our GAP) the short group authentication string is functionally independent of the information to be authenticated, whereas direct binding protocols require group authentication string to be dependent on the information devices want to authenticate. In direct binding protocols (e.g. GMA from [18]) this is achieved by making all group members agree on a short-output hash or digest of a complete description of the protocol run; this short-output digest is a short group authentication string. A common approach to generating a short-output digest in a provably secure way (in direct binding protocols) is to use universal hash functions. Note that the random keys required by the universal hash functions might be significantly longer than the hash output in several constructions of universal hash functions invented to date and these long keys have to be exchanged between the devices [28], [18]. At the same time in our GAP protocol we use and communicate only short random keys/nonces (in addition to information to be authenticated). It is this difference in the key lengths that makes indirect binding schemes more efficient in terms of a communication cost.

We compared the communication cost of the representative provably secure directly binding scheme proposed by Laur and Pasini [18] (the GMA protocol) and our protocol in Table 1. As shown in the table the advantage of our GAP protocol over GMA (expressed as the difference in the number of exchanged bits) is $M \cdot (|K_L|/2 - |N|)$ where $|K_L|/2$ represents a short-output hash or digest (as used in GMA), M being the group size and N is a short random key/nonce (as used in our GAP protocol). For example, with SHA-256 (thus $|K_L| = 512$ bits [18]) and $|N| = 15$ bits the advantage of GAP over GMA amounts to $241 \times M$ bits per device, i.e. for the whole group to $241 \times M^2$ (e.g. for $M = 30$ devices the difference is 26.5 KB in ideal conditions - no retransmissions).

For completeness, we also provide the comparison between the two protocols in terms of the computation

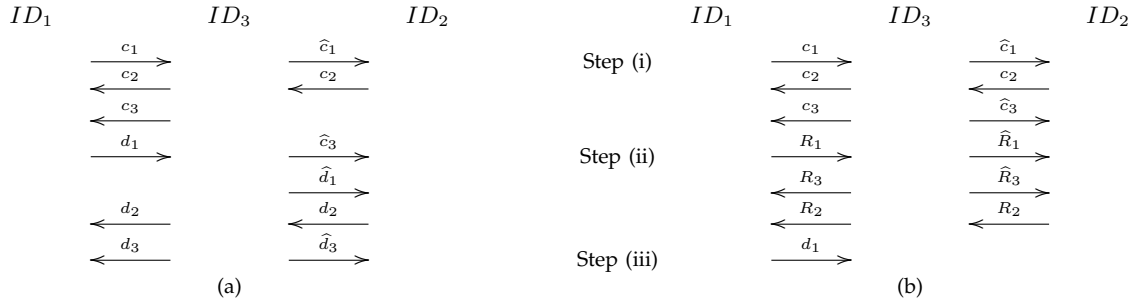


Fig. 3. (a) An example of the insider attack on GAP. Here the attacker ID_3 wants to impersonate itself as ID_1 to the sensor device ID_2. (b) Strengthening GAP.

cost (Table 1). For this we used the same computation cost model[2] as in [28]. Referring back to Table 1 note that $|K_L| = 2 \cdot length\big(h(\cdot)\big)$ [18]. It turns out that GAP has advantage over GMA also in terms of the computation cost, which is similar to the advantage in the communication cost.

4 Securing GAP Against Compromised Devices

Theorem 1 holds under the assumption that all nodes in $\mathcal{G}$ are trusted. In addition to the necessary requirement of providing security against the man-in-the-middle attacker, it is also desirable that the initialization provides protection against compromised nodes. As pointed out in [17], a manufacturer may sneak in malicious sensor node(s) along with normal sensor nodes shipped to a customer. Also, an adversary could insert his malicious code inside already deployed sensor network that requires keying with new devices brought to the field. Here we show how to strengthen the GAP to withstand insider attacks (compromised sensors).

4.1 Insider Attack on GAP

For simplicity, let us consider the following scenario. A user wants to initialize a total of 3 sensor devices (having ID_1, ID_2 ID_3). Let us assume that the device with ID_3 is compromised and controlled by the attacker. This device will try to impersonate itself to device ID_2 as ID_1. To accomplish this, ID_3 does the following (as shown in Figure 3(a)). It first blocks the commitment c_1 from ID_1 to ID_2 and replaces it with its own $\widehat{c}_1$ (and thus replacing N_1 with $\widehat{N}_1$ and PK_1 with its own $\widehat{PK}_1$). Later, ID_3 sends c_3 only to ID_1 in order to trigger it to open N_1. Now, ID_3 creates $\widehat{c}_3$ (in which it commits to $N_3 \oplus N_1 \oplus \widehat{N}_1$) and sends it to ID_2. It is easy to verify that at the end all the devices will share equal $GAS = N_1 \oplus N_2 \oplus N_3$. However, ID_3 successfully replaced ID_1's public key PK_1 with the one of its own choosing $\widehat{PK}_1$.

2. The computation cost is as follows: (1) $cost\big(h(m)\big) \approx length\big(h(\cdot)\big) \times length(m)$ and (2) $cost(commit(m)) \approx cost\big(h(m)\big)$.

4.2 Strengthening GAP Against Insider Attacks

To strengthen the basic GAP against insider attacks, we introduce an additional phase in the original protocol. More specifically, each device $i \in \mathcal{G}$ will generate a short (ℓ bits long, e.g. 15 bits) random number R_i and transmit it at the end of Phase II of the GAP over a radio channel. The purpose of this random number is to explicitly signify the completion of the Phase II on the side of the given device. The strengthened protocol is shown in Figure 4. Note that the Phase IV is similar to Phase III in the original GAP with the difference that instead of verifying the order in which messages are received, each device verifies random number $\widehat{R}_j$ (received in clear over a radio channel) against $\widehat{R}'_j$ extracted from the commit message $\widehat{c}_j$. *In this way, we not only mitigate the insider attack, but we also alow the devices to exchange the messages in an arbitrary order*. The cost of this solution is only ℓ bits (e.g. 15) per device. Although the messages are now exchanged in an arbitrary order the strengthened GAP remains secure (even against the attack introduced in Section 3.2), as we discuss next.

Let us consider again the scenario introduced in Section 4.1. Three nodes want to exchange some authentic information (e.g. public keys) using the strengthened GAP where the device ID_3 is compromised. As a part of the insider attack, this device will try to impersonate itself as ID_1 to ID_2 and replace public key PK_1 with one of its own choosing $\widehat{PK}_1$. In order to accomplish this the attacker blocks the commitment c_1 to the device ID_2 and replaces it with $\widehat{c}_1$ (therefore replacing N_1, R_1 and PK_1 by $\widehat{N}_1$, $\widehat{R}_1$ and $\widehat{PK}_1$) as shown in Figure 3(b) (step (i)). Next, the attacker sends its commitments c_3 and $\widehat{c}_3$ to the devices ID_1 and ID_2, respectively. The attacker has to make sure that $GAS_1 = GAS_2$ on devices ID_1 and ID_2, respectively. In the strengthened GAP, the devices ID_1 and ID_2 generate nonces N_1 and $\widehat{N}_1$ independently of each other, respectively. To generate $\widehat{N}_3$ (and send $\widehat{c}_3$) ID_2 has to trigger ID_1 to see N_1 (step (iii)). An adversary does not benefit from seeing decommitment d_1 when trying to construct $\widehat{c}_3$. Indeed, it follows that at the moment at which ID_1 revealed d_1, the device ID_1 must already have received the nonces R_2 and R_3 (step (ii)), as well as all the commitments. By sending the nonce R_i in the Phase III of the strengthened GAP, each device

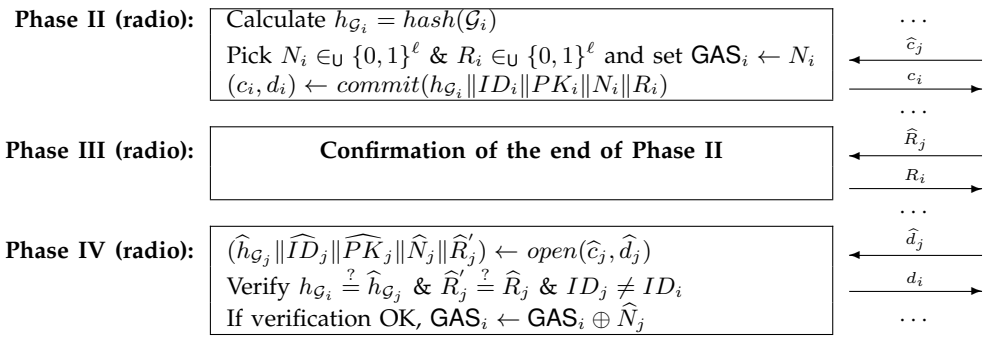
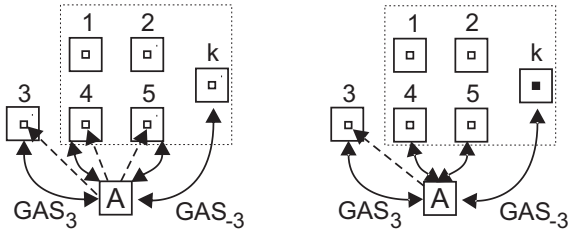


Fig. 4. Strengthening Group message Authentication Protocol (GAP) against insider attack. Phases I and V are not shown as they are identical to the first and the last phases in the original GAP.

$i \in \mathcal{G}$ acknowledges that it has successfully received the commitments $\widehat{c}_j$ in Phase II of the strengthened GAP from the devices $j \in \mathcal{G}$. In this way, the device i will send its decommitment d_i only after all the devices j previously acknowledged (with R_j) their successful reception of the commitments.

Manipulating the group size. The important security consideration of GAP is that the user is required to enter the group size into a single arbitrary sensor (e.g. a coordinator). However, a compromised coordinator could manipulate the entered group size. This is true for any protocol that falls in this category of protocols [22], [23], [11]. The simplest solution for this problem would be to assume that at least one device is not compromised and that user enters the group size into each device [23], [11].



GAS_{ki}	GAS_{-3i}	GAS_{3i}	Attack
1	0	0	yes
1	0	1	no
0	1	0	yes
0	1	1	yes

(c)

Fig. 5. The attacker A, with the aid of a laser, tries to (a) modify both GAS_3 and GAS_{-3} to match each other, and (b) modify only GAS_3 to match the fixed GAS_{-3}. The dashed arrows represent transmissions by the attacker using directed light source.

5 Securing a Visible Light Channel

In this section we first describe possible attacks on messages transmitted over a visible light channel when on-off keying modulation is used. We then show how to secure the transmission of a GAS over a semi-authenticated VLC.

5.1 Attacks on VLC and Preventive Mechanisms

5.1.1 No encoding of the GAS

In this scenario, the coordinator and the other sensor devices would simply transmit the GAS in its original form via VLC. Let us consider the scenario shown in Figure 5(a). Here, an attacker runs the Phases I-III of the GAP protocol and establishes two different group authentication strings, GAS_3 with device 3 and GAS_{-3} with the remaining devices. From Theorem 1, it follows $\mathbf{P}[\mathsf{GAS}_3 = \mathsf{GAS}_{-3}] \leq 2^{-\ell} + \epsilon$. If $\ell = 15$ or 20 bits, most likely GAS_3 and GAS_{-3} will differ with a high probability. Normally, this will be detected by the user in Phase IV of the GAP. However, in the semi-authenticated model of VLC, the adversary can flip bits 0 to 1 using a directed light source (e.g., a laser pointer). By flipping all the bits 0 to 1 in both GAS_3 and GAS_{-3} the user will see all 1s on all the sensor devices and wrongly conclude that the verification is successful. Please note that all 1s is a legitimate GAS.

5.1.2 Manchester and Berger Coding

We can mitigate the above bit flipping attack by using **Manchester** coding ($0 \rightarrow 01$ and $1 \rightarrow 10$). Manchester encoded GAS contains an equal number of 0s and 1s. To verify the GAS, a user would have to count the number of 0s and 1s in the transmitted sequence and also verify that the sequence has at most 2 subsequent bits set to 0 or 1. This solution doubles the size of GAS and requires the user to perform additional verifications (count the number of 0s and 1s, verify that the sequence has at most 2 subsequent 1s and 0s). Clearly, this is not an optimal option given that usability is severely deteriorated. In Figure 6(b) we compare different encodings wrt their impact on both usability and security. Clearly, the Manchester coding appears in the upper left corner

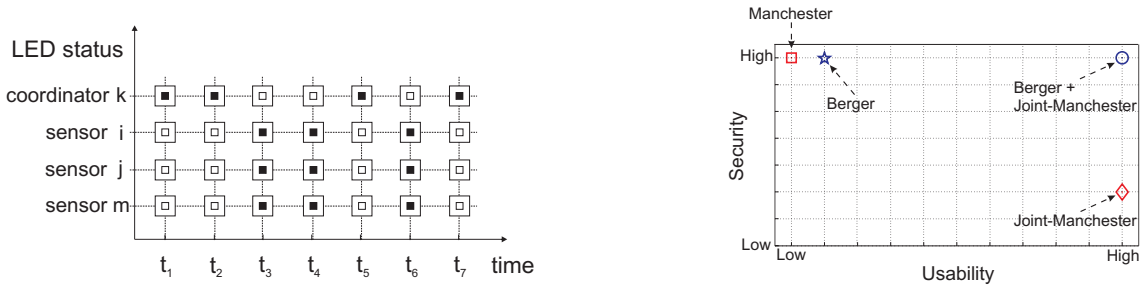


Fig. 6. (a) Joint Manchester coding for GAS=0011010. The LEDs on sensor devices i, j and m always occupy the opposite state of the coordinator k (a colored box indicates the LED is ON), (b) Security vs. Usability tradeoff of the proposed solutions for the GAS verification via VLC.

(marked with the square). In an attempt to increase the usability of the GAS verification procedure we could use Berger codes [6].

The **Berger** code is a well know unidirectional error detecting code. Berger codes can detect any unidirectional error in a given codeword. Unidirectional errors are errors that only flip zeros into ones or only ones into zeros, but not both at the same time. Let us consider a binary string (vector) $\mathbf{s}$ of size ℓ bits. Then a Berger coded string $\mathbf{s}$ (denoted $\mathsf{Ber}(\mathbf{s})$) is defined as follows: $\mathsf{Ber}(\mathbf{s}) \equiv \mathbf{s}\|\mathbf{s_B}$, where $\mathbf{s_B}$ represents (in binary) the number of zeros in $\mathbf{s}$. The Berger code appends to $\mathbf{s}$ the *check value* $\mathbf{s_B}$ of size $\lceil \log_2(\ell+1) \rceil$ bits, giving the Berger code of length $\ell + \lceil \log_2(\ell+1) \rceil$.

Example 1: For $\mathbf{s} = 1001101$, we have $\mathbf{s_B} = 011$ and $\mathsf{Ber}(\mathbf{s}) = 1001101\mathbf{011}$.

This coding is secure given that the user counted correctly the number of 0s in GAS, converted it to the binary representation and compared it successfully with the Berger check value. Clearly this is too heavy for an end user and therefore highly unusable. This places Berger coding next to Manchester coding in Figure 6(b).

5.1.3 "Joint-Manchester" Coding

We have seen that neither Manchester nor Berger coding result in a usable solution (Figure 6(b)). To improve the usability while trying to preserve the security, we introduce another coding scheme, termed "Joint-Manchester" coding. In this solution, the GAS is initially Manchester coded. However, each sensor device transmits only the half of the Manchester encoded GAS, according to the following rule: the coordinator and other sensor devices transmit even and odd bits of the Manchester encoded GAS, respectively. In Figure 6(a) we show an example of "Joint-Manchester" coding with coordinator k. Note that the devices other than k always share the same state.

The important difference from the usability point of view, with respect to regular Manchester coding, is that the user now has to only make sure that all the sensor devices other than the coordinator share the same LED state and is opposite of the LED state on the coordinator (please refer to Figure 6(a)). In Section 7, we show that "Joint-Manchester" coding can be easily verified by the user. This solution significantly decreases the time required to transmit the Manchester encoded GAS (to only ℓ bits, ℓ being the size of GAS). It does not put any additional effort on the user.

How secure is the "Joint-Manchester" coding? To answer this question, let us consider the scenario shown in Figure 5(b). Here, the attacker runs Phases I to III of the GAP protocol and establishes two likely different GAS values, namely, GAS_3 with device 3 and GAS_{-3} with the remaining devices. The goal of the attacker is to flip bits of GAS_3 (using a directed light source) such that the modified GAS_3 (denoted $\widehat{\mathsf{GAS}_3}$) satisfies $\widehat{\mathsf{GAS}_3} = \mathsf{GAS}_{-3}$. Note that the attacker has no advantage in flipping bits in GAS_{-3} and/or GAS_k as the devices other than the coordinator should occupy the state that is opposite to the one of the coordinator k. The table given in Figure 5(c) shows possible combinations of GAS_3 and GAS_{-3} (their ith bits) that are beneficial for the attacker. Thus, if the ith bits of GAS_{-3} and GAS_3 are equal, an attacker will not need to modify them in any way. On the other hand, if the ith bits of GAS_{-3} and GAS_3 equal 1 and 0, respectively, an attacker could flip $0 \rightarrow 1$ by using the laser. If the ith bits of GAS_{-3} and GAS_3 are 0 and 1, the attacker will be unable to flip $1 \rightarrow 0$ for he cannot switch OFF an already powered ON LED.

We conclude that 3 combinations out of 4 are beneficial to 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 $(3/4)^\ell$ as opposed to $2^{-\ell}$ (the probability of a successful attack where the attacker is unable to modify the GAS). If $\ell = 15$, the probability of a successful attack in a single attempt increases from 2^{-15} to approximately 2^{-6}. From this security analysis we conclude that the "Joint Manchester" coding is user-friendly but less secure than the basic Manchester coding. Therefore, in Figure 6(b) "Joint Manchester" coding appears in the bottom right corner.

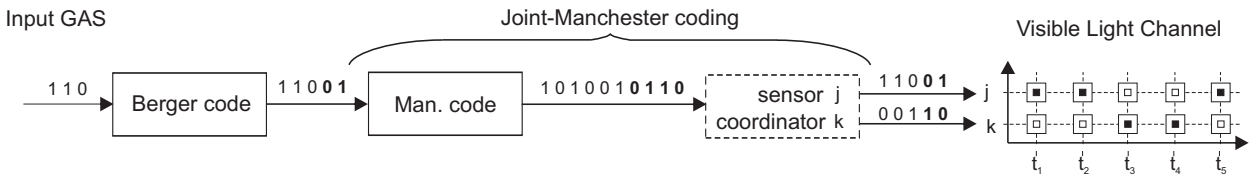


Fig. 7. An example of the GAS verification via VLC using Berger-Manchester encoding. Labels j and k stand for jth sensor device and the coordinator, respectively.

5.1.4 Berger-"Joint Manchester" Coding

Finally, we show that by combining "Joint Manchester" coding and Berger codes (Berger-Manchester coding from here on) we obtain a highly usable and yet a secure solution (Figure 6(b)). As shown in Figure 7 the GAS is first Berger encoded and then "Joint-Manchester" encoded. Recall, with "Joint-Manchester" coding each sensor device transmits only a half of the Berger-Manchester encoded GAS, according to the following rule: the coordinator and the other sensor devices transmit even and odd bits of the Manchester encoded GAS, respectively. This solution significantly decreases the time required to transmit the Manchester encoded GAS (to only $\ell + \lceil \log_2(\ell+1) \rceil$ bits, ℓ being the size of GAS). As in "Joint-Manchester" coding, Berger-Manchester coding does not require the user to perform any additional task. Therefore, in Figure 6(b) Berger-Manchester coding appears in the upper right corner. The price that we have to pay for the increased security is the increased number of bits that the user has to verify by $\lceil \log_2(\ell+1) \rceil$ bits (due to the Berger check value). In Section 7 we show that Berger-Manchester coding can be easily verified by the user. We next prove the security of the Berger-"Joint-Manchester" coding.

Fact 1: Let $i, j \in \mathcal{G}$ be any two sensor devices such that $\mathsf{GAS}_i \neq \mathsf{GAS}_j$. Then, the group authentication string GAS_k as generated by the coordinator, satisfies: $(\mathsf{GAS}_k \neq \mathsf{GAS}_i) \vee (\mathsf{GAS}_k \neq \mathsf{GAS}_j)$.
In other words, GAS_k cannot be equal to the respective GAS values of both the sensor device i and the device j. Therefore, to detect an error (i.e., a potential attack) in the initialization process, it is sufficient to compare the GAS of the coordinator with the GAS value of each remaining sensor device.

Fact 2: Let $a, b \in \mathbb{N}_0$ be two natural numbers (including zero) such that $a > b$, and let $\mathbf{a}, \mathbf{b} \in \{0,1\}^\ell$ be their binary representations (vectors). Then, $\exists i \in \{0, 1, \ldots, \ell-1\}$ such that $a_i > b_i$.

We denote with $\mathsf{Man}(\mathbf{a})$ the Manchester encoded binary vector $\mathbf{a} \in \{0,1\}^\ell$. We also use notation $\mathbf{a_{Odd}}$ and $\mathbf{a_{Even}}$ to denote the odd and even bits of a Manchester-Berger encoded vector $\mathbf{a}$, respectively. For example, $\mathbf{a} = 1001101$, $\mathbf{a_{Odd}} \leftarrow [\mathsf{Man}(\mathsf{Ber}(\mathbf{a}))]_{\mathbf{Odd}}$ and $\mathbf{a_{Even}} \leftarrow [\mathsf{Man}(\mathsf{Ber}(\mathbf{a}))]_{\mathbf{Even}}$ we obtain: $\mathbf{a_{Odd}} = 1001101011$ and $\mathbf{a_{Even}} = 0110010100$. Finally, we use $\mathbf{1}$ as a shorthand notation for the vector comprising all ones (the vector length should be clear from the context) and $\oplus$ to denote bitwise addition modulo 2. Next, we state our main result in this section.

Theorem 2: Let $\mathbf{a}, \mathbf{b} \in \{0,1\}^\ell$ be two arbitrary but different ℓ-bit binary vectors ($\mathbf{a} \neq \mathbf{b}$). It is not possible to modify $\mathbf{a_{Odd}} \leftarrow [\mathsf{Man}(\mathsf{Ber}(\mathbf{a}))]_{\mathbf{Odd}}$ and $\mathbf{b_{Even}} \leftarrow [\mathsf{Man}(\mathsf{Ber}(\mathbf{b}))]_{\mathbf{Even}}$ using only unidirectional changes of the type $0 \rightarrow 1$ such that the resulting binary vectors, denoted $\widehat{\mathbf{a}}_{\mathbf{Odd}}$ and $\widehat{\mathbf{b}}_{\mathbf{Even}}$, respectively, satisfy $\widehat{\mathbf{a}}_{\mathbf{Odd}} \oplus \widehat{\mathbf{b}}_{\mathbf{Even}} = \mathbf{1}$.

Proof: Let us denote with $\mathsf{Hw}(\cdot)$ the Hamming weight of a given binary vector. Considering the binary vectors $\mathbf{a}, \mathbf{b} \in \{0,1\}^\ell$ for which $\mathbf{a} \neq \mathbf{b}$, we can distinguish the following three cases:

1. $\mathbf{a} \neq \mathbf{b}$ and $\mathsf{Hw}(\mathbf{a}) = \mathsf{Hw}(\mathbf{b})$
2. $\mathbf{a} \neq \mathbf{b}$ and $\mathsf{Hw}(\mathbf{a}) < \mathsf{Hw}(\mathbf{b})$
3. $\mathbf{a} \neq \mathbf{b}$ and $\mathsf{Hw}(\mathbf{a}) > \mathsf{Hw}(\mathbf{b})$.

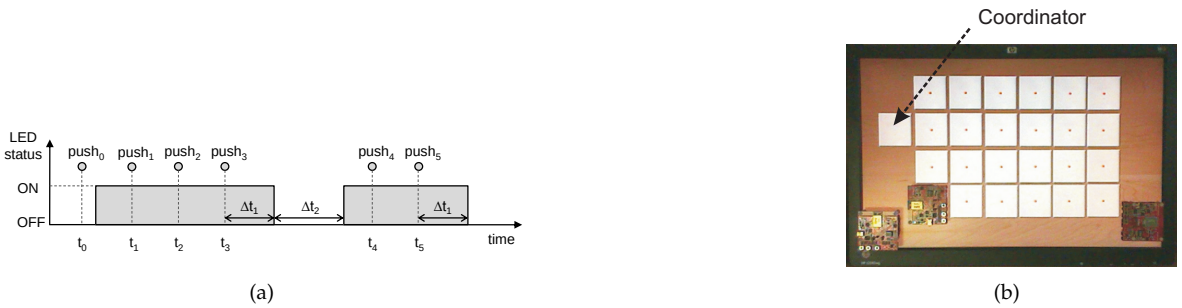
Case 1. Let $\mathbf{a} \neq \mathbf{b}$ and $\mathsf{Hw}(\mathbf{a}) = \mathsf{Hw}(\mathbf{b})$. If two binary vectors are not equal but have the same Hamming weight (i.e., $\mathbf{a_B} = \mathbf{b_B}$), then clearly $\exists i \in \{0, 1, \ldots, \ell-1\}$ such that $a_i = 1$ and $b_i = 0$. From the definitions of $\mathbf{a_{Odd}}$ and $\mathbf{b_{Even}}$ we have $a_{\mathbf{Odd},i} \leftarrow [\mathsf{Man}(a_i)]_{\mathbf{Odd}} = [(1,0)]_{\mathbf{Odd}} = 1$ and $b_{\mathbf{Even},i} \leftarrow [\mathsf{Man}(b_i)]_{\mathbf{Even}} = [(0,1)]_{\mathbf{Even}} = 1$. Since only unidirectional changes ($0 \rightarrow 1$) are allowed, it is not possible to modify neither $a_{\mathbf{Odd},i}$ nor $b_{\mathbf{Even},i}$. Therefore, $\widehat{a}_{\mathbf{Odd},i} = \widehat{b}_{\mathbf{Even},i}$ and consequently $\widehat{\mathbf{a}}_{\mathbf{Odd}} \oplus \widehat{\mathbf{b}}_{\mathbf{Even}} \neq \mathbf{1}$.

Case 2. Let $\mathbf{a} \neq \mathbf{b}$ and $\mathsf{Hw}(\mathbf{a}) < \mathsf{Hw}(\mathbf{b})$. From $\mathsf{Hw}(\mathbf{a}) < \mathsf{Hw}(\mathbf{b})$ it follows that $\mathbf{a}$ has a larger number of zeros than $\mathbf{b}$ and hence the larger Berger check value, i.e., $a_B > b_B$ (in base-10 notation). Since both a_B and b_B are from $\mathbb{N}_0$, it follows from Fact 2 that $\exists i \in \{0, 1, \ldots, \lceil \log_2(\ell+1) \rceil - 1\}$ such that $a_{Bi} = 1$ and $b_{Bi} = 0$. Now, using the same reasoning as in the first case (**Case 1**), it follows that in this case, too, we have $\widehat{\mathbf{a}}_{\mathbf{Odd}} \oplus \widehat{\mathbf{b}}_{\mathbf{Even}} \neq \mathbf{1}$.

Case 3. Let $\mathbf{a} \neq \mathbf{b}$ and $\mathsf{Hw}(\mathbf{a}) > \mathsf{Hw}(\mathbf{b})$. From $\mathsf{Hw}(\mathbf{a}) > \mathsf{Hw}(\mathbf{b})$ it follows directly that $\exists i \in \{0, 1, \ldots, \ell-1\}$ such that $a_i = 1$ and $b_i = 0$. Therefore, following the same steps as in the first case (**Case 1**), we can conclude that in this case, too, we have $\widehat{\mathbf{a}}_{\mathbf{Odd}} \oplus \widehat{\mathbf{b}}_{\mathbf{Even}} \neq \mathbf{1}$.

Thus, in all the possible cases we have $\widehat{\mathbf{a}}_{\mathbf{Odd}} \oplus \widehat{\mathbf{b}}_{\mathbf{Even}} \neq \mathbf{1}$ (given $\mathbf{a} \neq \mathbf{b}$ and the unidirectional changes $0 \rightarrow 1$). □

In other words, the Berger-Manchester coding is secure in the model where the attacker can only flip bits



(a) (b)

Fig. 8. (a) An example of the group size verification for $M = 32$ devices. Experimental setup: (b) a 22-inch monitor (placed horizontally) featuring 25 sensor devices.

0 into 1 on a visible light channel (semi-authenticated channel).

6 Implementation Details

In this section we discuss some implementation aspects of the proposed GAP based secure initialization mechanism, which are relevant for the usability study that we conduced. More precisely, we describe a simple method for entering the group size into the coordinator (Phase IV of GAP). Note also that the user needs to know the status of each device through the initialization process. In our implementation, this is accomplished using the sensor's LED; due to lack of space, we do not provide further details of this implementation.

Entering the Group Size and Synchronization. It is essential for the security of GAP that at least one device, from the group of devices being initialized, knows the correct group size. The procedure goes as follows. Let us assume that the user wishes to initialize $M < 100$ devices. We can represent M using decimal notation as follows: $M = M_1||M_2$, where $M_i \in \{0, 1, ..., 9\}$ (for example, if $M = 32$, then $M_1 = 3$ and $M_2 = 2$). Next, the user takes the coordinator (indicated with the LED powered OFF; other devices blink) and initiates the procedure for entering the group size with a short push on the button ($push_0$ in Figure 8(a)). In turn, the coordinator's LED powers ON which indicates to the user that the coordinator is ready to accept the first digit of M ($M_1 = 3$ in our example). To enter the first digit the user pushes the button M_1 times ($push_1$, $push_2$ and $push_3$ in Figure 8(a)). After that, the user waits the predefined time period Δt_1 for the LED to blink once (the LED powers subsequently OFF and ON as shown in Figure 8(a)). This blink indicates to the user that the coordinator is ready to accept the second digit of M (i.e., M_2). Again, the user enters the second digit by pushing the button M_2 times ($push_4$ and $push_5$) and waits the predefined time period Δt_1 (Figure 8(a)). After the coordinator "concludes" that the user has entered the second digit, the coordinator assembles the group size by concatenating two digits and compares the result with the group size that the coordinator has learned from the Phase I of the GAP. If the two match, the coordinator advances to the Number OK state (continuous LED blinking). Otherwise, the coordinator enters the Error state (constantly powered LED ON).

Synchronization. The coordinator initiates simultaneous and synchronized transmission of Berger-Manchester encoded GAS over VLC on all devices. The synchronization can be achieved by having the coordinator send SYNC messages over a radio channel to the other devices. Any attempt of jamming or injecting synchronization messages will result in desynchronization among sensor devices. This is eventually detected by the user because the sensor's LED blinks fast in all states but during the GAS transmission. Indeed, the fast blinking of a LED will overlap (in time) with much slower GAS transmission; in our implementation a single LED pulse during the GAS transmission is 4 seconds long. In future, we plan to study these aspects in greater detail. A similar approach to synchronizing GAS transmissions appears in Prasad and Saxena [32].

7 Usability Evaluation

Experimental setup. Our focus in this preliminary study was to verify the thesis that Berger-Manchester coding (the GAS verification) is easy to interpret (perform) for an end user. In addition we evaluated the procedure for entering the group size into the coordinator (Figure 8(b)). For this purpose we implemented a simple simulator called BlinkTest (Figure 8). The BlinkTest allows us to simulate different scenarios in which sensor devices are placed on arbitrary virtual surfaces (e.g., a desk office as shown in Figure 8(b)). In our study we arranged sensor devices in four rows and six columns; this is logical decision when working with a large number of devices. BlinkTest also allows us to choose different casings for sensor devices. In this study a simplistic white casing with one red LED (Figure 8) was used. As can be seen from Figure 8(b), the size of a virtual sensor device matches the one of a real ZigBee sensor device (6x6 cm^2). In BlinkTest we can configure virtual sensor to blink arbitrary GAS values in synchrony. A user in our study interacts with the simulator via the 22-inch monitor that is placed horizontally on a office desk (Figure 8). The user uses a mouse to select the coordinator (Figure 8(b));

TABLE 2
The testers' demographic info as well as computer and mobile devices usage.

Age		Sex		Eyesight		Using Computer (hours/week)			
18-25	over 25	Male	Female	No glasses/contacts	Glasses/contacts	> 5	6-15	15-30	< 30
28	0	22	6	18	10	1	8	11	8
Using Internet (hours/week)				**Using mobile device pairing**		**Feel secure while using wireless**			
> 5	6-15	15-30	< 30	Y	N	Agree	Do not agree	Neutral	Don't know
5	11	11	1	24	4	15	3	7	3

the left mouse click simulates the pressing of a sensor's button.

Test cases. We first tested the users' capability to correctly enter the group size into the coordinator. We used the following group sizes: 7, 10, 16 and 25 sensor devices. In the second round of usability tests we studied the ability of users to perform GAS verification and their ability to detect intentionally introduced errors. In these tests the blinking period was set to 4 seconds. The length of the GAS as transmitted over visible light channel was 19 bits.

We created eleven test cases for the GAS verification, which we divide into four categories: (1) GAS mismatch between the coordinator and a single sensor node in: (1.1) the first bit, (1.2) a middle bit and (1.3) the last bit, (2) GAS mismatch between the coordinator and all sensor devices in: the (2.1) first bit, (2.2) a middle bit and (2.3) the last bit and (3) GAS mismatch between the coordinator and all the remaining sensor devices but one in: (3.1) the first bit, (3.2) a middle bit and (3.3) the last bit. At the end, we tested the scenario with (4) no GAS mismatches. The tests involved a 25 sensor devices (Figure 8).

Procedure. A total of 28 participants took part in the usability study. The testers were given a short introduction to the initialization procedure, which involved the description of the node's state diagram, possible applications of such a pairing scenario (access points, ad-hoc networks, smart homes etc.). None of the participants have taken part in any of our tests before. All the participants were in their early twenties. Table 2 summarizes the participants' demographic information as well as information about their everyday usage of computers and mobile devices. The usability test is divided into two phases: a training phase and a testing phase. The training phase served the purpose of teaching the participants (i) how to enter the group size on the coordinator and (ii) how to perform the GAS verification on sensor devices. The training phase lasted for about 5 minutes. In the testing phase the participants performed the actual test. At the end of every usability test, the participants completed a post-test questionnaire, which involved the System Usability Scale [7] to numerically express their subjective opinion about the usability of the tested procedures.

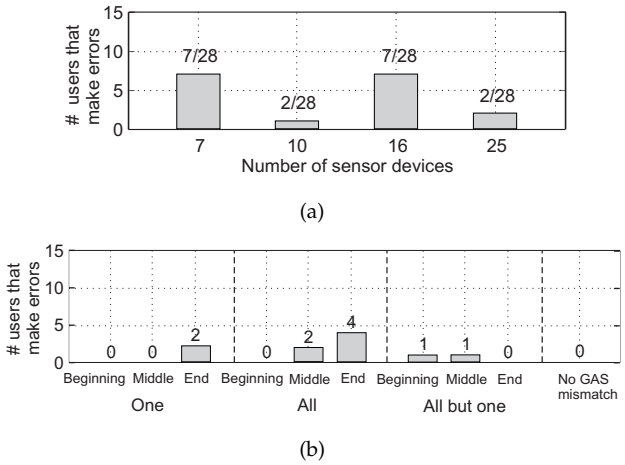


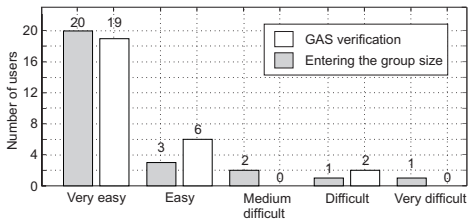
Fig. 9. Testers that made mistake: (a) while entering the group size, (b) for particular GAS verification test case.

Results of the Study

Each of the 28 participants performed 4 test cases for testing the purpose of the procedure for entering the group size and 10 test cases for testing the GAS verification procedure, leading to a total of 392 test cases.

Entering the group size. Each user was asked to enter once each of the following numbers 7, 10, 16 and 25, while using the procedure presented in Section 6 (Figure 8(a)). As the results in Figure 9(a) show, some users experienced problems while entering 7 and 16. In the first case the users had to enter two digits: 0 and 7. It turned out that the users would miss completely to enter the first digit 0. The high error rate with the group size of 16 is due to the fact the users confused the $push_0$ event in Figure 8(a) with the $push_1$. The average time for entering the group size was around 16, 14, 17 and 18 seconds for 7, 10, 16 and 25 devices, respectively. Please note that these included times Δt_1 and Δt_2 in Figure 8(a). Evaluating a usable-security application with young and educated participants is a natural first step (an application that does not fare well with them is unlikely to be acceptable by other samples of population), and therefore, our study only represents a preliminary evaluation. Future work is needed to evaluate the method with a sample representative of a larger population.

GAS verification via VLC. The testers were asked to observe the sensor devices on the display as shown in Figure 8, and to indicate (through a keyboard) if and when the status of the coordinator's LED (the isolated sensor device in Figure 8(b)) is incompatible with the LED status on the other sensor devices. Recall that by the Berger-"Joint Manchester" coding the status of the LED on the coordinator must always be opposite of the status of the LED on all the other sensor devices (Section 5.1.4). Each test case included the initialization of 25 sensor devices (Figure 8(b)). In Figure 9(b) we plot the number of testers that make mistakes for different



(a)

(b)

Fig. 10. (a) User feedback on the usability of the initialization protocol. (b) Zero-configuration auxiliary device: Using a smart phone equipped with a camera to assist the initialization of a larger number sensor devices.

GAS verification scenarios. Overall, we see a reasonably low error rates. As expected, we observe the highest error rate when GAS mismatch occurs at the end of the GAS transmission (2 errors in the scenario when only one device has incompatible GAS with the coordinator and 4 errors in the scenario when all the devices have incompatible GAS values with the coordinator). The reason for this is that the users become less focused towards the end of the GAS verification phase. Somewhat higher error rate in the scenario where all the sensors have incompatible GAS with the coordinator is also understandable, as in this case all the sensor devices share the same LED state; it is much easier to detect the incompatible GAS on only one device. Finally, from the last test case (No GAS mismatch) we conclude that there were no false positives in our study. Note that the duration of the GAS verification phase can be calculated by multiplying the duration of the LED pulse (4 seconds in our implementation) with the size of the encoded GAS (19 bits in our case). In our study this amounts to approximately 80 seconds. Given that the user initializes 25 sensor devices this amounts to around 3.2 seconds per device.

Questionnaire. At the end of the usability tests the users were asked to fill in the post-test questionnaire from which the System Usability Score (SUS) [7] was calculated. The average SU-score for 28 users was 80,8 (out of 100). Finally, Figure 10(a) summarizes the users' answers on the questions related to difficulty of the procedure for entering the group size and detection of mismatches in the GAS verification phase. As shown, most of the users found these two procedures relatively easy to use.

Improving Usability and Scalability with a Zero-Configuration Auxiliary Device. In some situations the user may want to initialize even larger number of nodes than we considered in this paper (e.g. more than 100 in several batches). Note that there is a certain limit to the number of devices that can be initialized in one batch because of the constrained nature of the devices as well as that of the human operator. Therefore, to significantly improve scalability, usability and reduce likelihood of errors we can use a camera on a smartphone if available, to record a group authentication string (GAS) transmitted using LEDs, as shown in Figure 10(b). Most existing solutions that involve cameras, such as [11], [27], [36], [37], [35], [30], [22], [41] require video processing or pattern recognition services installed and preconfigured with the camera devices. On the contrary, in the case of our GAP protocol, the GAS verification aided by a smartphone requires no special services or configuration on the side of the smartphone. All that is required from the user is to record a the GAS procedure (Figure 10(b)) and review it as many times as necessary to make sure that LED of the group members at all times occupy the opposite state than the coordinator device. We have shown before that this is an easy task for the user thanks to the Berger-Manchester coding.

8 Related Work

Many existing key (pre-)distribution schemes for wireless networks rely on unspecified secure key initialization mechanisms. Here, we overview existing initialization mechanisms.

In Resurrecting Duckling [39], a physical contact is required to securely establish a secret key. It requires specialized hardware and may not scale well. Similarly, Talking to Strangers [4] 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 [27]. Key authenticity is achieved through certified public keys.

In Shake Them Up [9], user establishes a secret key between two nodes by holding and shaking the devices together while they send identical packets over the radio. This scheme may be violated by using radio fingerprinting. The three related schemes are Are You With Me [21], Smart-Its Friends [14] and [25]. Mayrhofer and Welch [26] also use an out-of-band laser channel constructed with off the shelf components for transmitting short authentication strings. According to [26], the proposed solution does not ensure complete authenticity of the the laser channel. Roman and Lopez [33] discuss general aspects of communication over a visible light channel.

In Key Infection [3], 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. In Message In a Bottle [17] and KALwEN [20], keys are sent in the clear to the nodes located inside a Faraday cage (a specialized hardware) that ensures key secrecy and authenticity. However, the number of simultaneously initialized nodes determines the size of the Faraday cage. In On-off Keying, the presence of an RF signal represents a binary 1, while its absence represents a binary 0 [8]. By using an unidirectional encoding scheme, On-off Keying ensures that an attacker is unable to modify a packet during transmission.

In the paper, Wong and Stajano [41] present device pairing and group key agreement multichannel protocols that use communication over a radio and an out-of-band channel (e.g. visual). However, their protocol requires each device to be capable of demodulating signals received over an OoB channel (i.e., they have to be equipped with a camera). In HAPADEP [38] 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 set of papers, Saxena and Uddin [35], [36], Saxena et. al. [37] and Perkovic et. al. [30] present device pairing methods based on devices equipped with LEDs and a video camera as the receiver. Li et. al. [22] also propose a protocol for the initialization of the large number of sensor devices that can be operated by a human. However, their protocol is insecure in the attacker model where an adversary performs flipping attacks in semi-authentic VLC.

In GAnGS [11] and SPATE [23] protocols for the secure exchange of authenticated messages among a group of N users are proposed. While GAnGS requires $O(N)$ interactions to authenticate the exchanged data, in SPATE each group member carries out N comparisons in parallel to authenticate other members' data.

9 Conclusion

We made several contributions in this paper. We proposed a novel multichannel protocol, called *Group message Authentication Protocol* (GAP), for user-friendly initialization of multiple resource-constrained wireless devices. The proposed protocol has minimal hardware requirements on the wireless devices: one LED and one button. Moreover, as an indirect binding scheme [28] GAP has a lower communication cost compared to existing *direct* binding protocols. GAP involves communication over a bidirectional radio channel and an unidirectional out-of-band visible light channel. The proposed protocol is shown to be secure in the very strong attacker model, where an attacker can eavesdrop, jam and modify transmitted messages on both the radio and the visible light channel. We also introduced a novel coding scheme (Berger-Manchester combination) for the secure communication over semi-authentic Visible Light Channel. Finally, we demonstrated the feasibility of the proposed initialization method via the usability study that indicates that the method has reasonably low execution time, minimal error rate and is user-friendly.

References

[1] 6LowPAN - RFC 4919. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. http://tools.ietf.org/html/rfc4919, last access, January, 2011.

[2] Mica2 Specifications. https://www.eol.ucar.edu/rtf/facilities/isa/internal/CrossBow/DataSheets/mica2.pdf, last access, January, 2011.

[3] R. Anderson, H. Chan and A. Perrig. Key Infection: Smart Trust for Smart Dust. In *IEEE International Conference on Network Protocols*, 2004.

[4] D. Balfanz, D. K. Smetters, P. Stewart and H. C. Wong. Talking to Strangers: Authentication in Ad-hoc Wireless Networks. In *Symposium on Network and Distributed Systems Security*, 2002.

[5] M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *CRYPTO*, 1993

[6] J. M. Berger. A Note on an Error Detection Code for Asymmetric Channels. Information and Control, 1961.

[7] J. Brooke. SUS: A Quick and Dirty Usability Scale. In *Usability Evaluation in Industry*, London, 1996.

[8] M. Cagalj, S. Capkun and J.-P. Hubaux. Key Agreement in Peer-to-Peer Wireless Networks. In *Proceedings of the IEEE Special Issue on Cryptography and Security*, 2006.

[9] C. Castelluccia and P. Mutaf. Shake Them Up!: A Movement-based Pairing Protocol for CPU-constrained Devices. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2005.

[10] H. Chan, A. Perrig and D. Song. Random Key Predistribution Schemes for Sensor Networks. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2003.

[11] C.-H. Owen Chen, C.-W. Chen, C. Kuo, Y.-H. Lai, J. M. McCune, A. Studer, A. Perrig, B.-Y. Yang and T.-C. Wu. GAnGS: Gather, Authenticate 'n Group Securely. In *Proceedings of the 14th Annual International Conference on Mobile Computing and Networking, MOBICOM*, 2008.

[12] W. Du, J. Deng, Y. S. Han and P. K. Varshney. A Pairwise Key Pre-Distribution Scheme for Wireless Sensor Networks. In *ACM CCS*, 2003.

[13] L. Eschenauer and V. D. Gligor. A Key-Management Scheme for Distributed Sensor Networks. In *ACM CCS*, 2002.

[14] 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 *International Proceedings of the 3rd International Conference on Ubiquitous Computing*, 2001.

[15] C. Karlof, N. Sastry and D. Wagner. TinySec: a Link Layer Security Architecture for Wireless Sensor Networks. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, 2004.

[16] A. Kumar, N. Saxena, G. Tsudik and E. Uzun. Caveat Emptor: A Comparative Study of Secure Device Pairing Methods. In *International Conference on Pervasive Computing and Communications (PerCom)*, 2009.

[17] C. Kuo, M. Luk, R. Negi and A. Perrig. Message-In-a-Bottle: User-Friendly and Secure Key Deployment for Sensor Nodes. In *Proceedings of the 5nd International Conference on Embedded Networked Sensor Systems (SenSys)*, 2007.

[18] S. Laur and S. Pasini. SAS-Based Group Authentication and Key Agreement Protocols. In *PKC, 11th International Workshop on Practice and Theory in Public-Key Cryptography*, 2008.

[19] S. Laur and S. Pasini. User-Aided Data Authentication. International Journal of Security and Networks, 4, 2009.

[20] Y.W. Law and G. Moniava and Z. Gong and P.H. Hartel and M. Palaniswami. KALwEN: A New Practical and Interoperable Key Management Scheme for Body Sensor Networks, TR-CTIT-08-67, 2008.

[21] 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*, 2004.

[22] M. Li, S. Yu, W. Lou and K. Ren. Group Device Pairing Based Secure Sensor Association and Key Management for Body Area Networks. In *IEEE INFOCOM*, 2010.

[23] Y.-H. Lin, A. Studer, H.-C. Hsiao, J. M. McCune, K.-H. Wang, M. Krohn, P.-L. Lin, A. Perrig, H.-M. Sun and B.-Y. Yang. SPATE: Small-group PKI-less Authenticated Trust Establishment. In *Proceedings of the MobiSys*, 2009.
[24] D. Liu and P. Ning. Establishing Pairwise Keys in Distributed Sensor Networks. In *ACM CCS*, 2003.
[25] R. Mayrhofer and H. Gellersen. Shake Well Before Use: Two Implementations for Implicit Context Authentication. In *Ubicomp*, 2007.
[26] R. Mayrhofer and M. Welch. A Human-Verifiable Authentication Protocol Using Visible Laser Light. In *International Conference on Availability, Reliability and Security*, 2007.
[27] J. M. McCune, A. Perrig and M. K. Reiter. Seeing-Is-Believing: Using Camera Phones for Human-Verifiable Authentication. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2005.
[28] L. H. Nguyen and A. W. Roscoe. Authentication Protocols Based on Low-bandwidth Unspoofable Channels: A Comparative Survey. Cryptology ePrint Archive, Report 2010/206.
[29] R. Nithyanand, N. Saxena, G. Tsudik and E. Uzun. Groupthink: Usability of Secure Group Association for Wireless Devices. In *Proceedings of the 12th ACM International Conference on Ubiquitous Computing*, 2010.
[30] T. Perkovic, I. Stancic, L. Malisa and M. Cagalj. Multichannel Protocols for User-Friendly and Scalable Initialization of Sensor Networks. In *5th Int. ICST Conference on Security and Privacy in Comm. Networks (Securecomm)*, 2009.
[31] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen and D. E. Culler. SPINS: Security Protocols for Sensor Networks. Wireless Networks, 8(5), 2002.
[32] R. Prasad and N. Saxena. Efficient Device Pairing Using "Human-Comparable" Synchronized Audiovisual Patterns. In *Applied Cryptography and Network Security (ACNS)*, 2008.
[33] R. Roman and J. Lopez. KeyLED - Transmitting Sensitive Data Over Out-of-Band Channels in Wireless Sensor Networks. In *Proceedings of the 4th IEEE International Workshop on Wireless and Sensor Networks Security (WSNS)*, 2008.
[34] P. Rost and G. Fettweis. On the Transmission-Computation-Energy Tradeoff in Wireless and Fixed Networks. CoRR, abs/1008.4565, 2010.
[35] N. Saxena and Md. B. Uddin. Automated Device Pairing for Asymmetric Pairing Scenarios. In *Proceedings of the 10th International Conference on Information and Communications Security*, 2008.
[36] N. Saxena and Md. B. Uddin. Blink 'Em All: Scalable, User-Friendly and Secure Initialization of Wireless Sensor Nodes. In *8th International Conference of Cryptology and Network Security, CANS*, 2009.
[37] N. Saxena, Md. B. Uddin and J. Voris. Universal Device Pairing Using an Auxiliary Device. In *Proceedings of the 4th Symposium on Usable Privacy and Security (SOUPS)*, 2008
[38] C. Soriente, G. Tsudik and E. Uzun. HAPADEP: Human-Assisted Pure Audio Device Pairing. In *Proceedings of the 11th International Conference on Information Security (ISC)*, 2008.
[39] F. Stajano and R. Anderson. The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. In *Proceedings of the 7th International Workshop on Security Protocols*, 2000.
[40] S. Vaudenay. Secure Communications Over Insecure Channels Based on Short Authenticated Strings. In *CRYPTO*, 2005.
[41] F. L. Wong and F. Stajano. Multichannel Security Protocols. In *IEEE Pervasive Computing, Special Issue on Security and Privacy*, 6(4), 2007.

Toni Perković received the Dipl. Ing. degree in telecommunications and electrical engineering from the University of Split, Croatia, in 2007. He is currently working toward the Ph.D degree at the Faculty of Electrical Engineering, Mechanical Engineering, and Naval Architecture (FESB), University of Split, Croatia. His research interests include the usability, design and analysis of security protocols for wireless networks, the usability and design of the secure authentication protocols. He is a member of the IEEE.

Mario Čagalj is an Associate Professor in the Department of Electronics at Faculty of Electrical Engineering, Mechanical Engineering, and Naval Architecture (FESB), University of Split, Croatia. He received the Dipl.Ing degree in computer science and electrical engineering from the University of Split, Croatia, in 1998, and the PhD degree in communication systems from the Ecole Polytechnique Federale de Lausanne (EPFL) in 2006. In 2000 and 2001, he completed the Predoctoral School in Communication Systems, EPFL. From 2001 to 2006, he was a research assistant in the Laboratory for Computer Communications and Applications (LCA) at EPFL. In December 2006, Mario Čagalj was elected Assistant Professor and in September 2010 he was promoted to Associate Professor at the University of Split, Croatia. His research interests include the design and analysis of security protocols for wireless networks, applied cryptography, applications of game theory to wireless (and wired) networks, and the design of energy-efficient communication protocols for wireless networks.

Toni Mastelić received the B.S. degree in computer science from the University of Split, Croatia, in 2009. During his study he worked as a student assistant at FESB, where he has also been engaged in several student as well as scientific research projects in computer and network security field. At the moment, Toni has been accepted as Erasmus exchange student at TU Wien, Austria, where he will be working toward his M.S. degree.

Nitesh Saxena is an Assistant Professor in the Department of Computer Science and Engineering at Polytechnic Institute of New York University (formerly Polytechnic University). He works in the areas of computer and network security, and applied cryptography. Nitesh obtained his Ph.D in Information and Computer Science from UC Irvine. He holds an M.S. in Computer Science from UC Santa Barbara, and a Bachelors degree in Mathematics and Computing from the Indian Institute of Technology, Kharagpur, India. Nitesh's Ph.D. dissertation on "Decentralized Security Services" has been nominated for the ACM Dissertation Award 2006. He is the recipient of the Best Student Paper Award at the Applied Cryptography and Network Security (ACNS) conference 2006.

Dinko Begušić received the B.S. degree in electrical engineering from the University of Split, Croatia in 1983, and the M.S. and Ph.D degrees in electrical engineering from the University of Zagreb, Croatia, in 1988 and 1992, respectively. Since 1985, he has been with the University of Split, Croatia, where he is currently a Professor and the chair of the communication technologies and signal processing at the Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture (FESB). From September 1990 to June 1991, he was a visiting researcher with the Universite Libre de Bruxelles, Bruxelles, Belgium. From February through July 1992, he was visiting researcher with the Kings College London, London, UK. From September 1997 until May 1998, he was with the University of Texas at Dallas, Richardson, TX, as a Visiting Assistant Professor. His research interests include communication systems and networks, digital signal processing for communications and adaptive algorithms. Dr.Begusic is a Co-Chair of the Conference on Software, Telecommunications and Computer Networks SoftCOM. He is a member of the IEEE.

APPENDIX - SKETCH OF THE PROOF OF THEOREM 1

A user wishes to initialize a set of M sensor devices. We denote this group with $\mathcal{G}$, i.e., $\mathcal{G} = \{ID_1, ID_2, \ldots, ID_M\}$, ID_i being the identity of the ith device (all IDs are unique). In the proof, we assume that the adversary does not belong to the set $\mathcal{G}$ (i.e., no sensor device from $\mathcal{G}$ is compromised). We further assume that each device has an access to a perfect random number generator. We denote with k the coordinator sensor device. Our security proof is based on the notion of *matching conversations* introduced by [5]. Informally, we say that devices $i, j \in \mathcal{G}$ ($i \neq j$) have matching conversations, if for each message m_i (m_j) sent out by i (j) at the time instant t_i (t_j), the device j (i) received the same (unaltered) message $m_{ij} = m_i$ ($m_{ji} = m_j$) at the time instant t_{i+1} (t_{j+1}), where $t_i < t_{i+1}$ ($t_j < t_{j+1}$). Clearly if all pairs of devices from $\mathcal{G}$ have matching conversations, then all messages transmitted must have arrived at intended destinations unaltered. In other words, the messages are authentic.

We say that all sensor devices from $\mathcal{G}$ "Accept" (shortly "All accept") if all the verifications in the GAP protocol (Figure 2) are successful. Let us further define an event S as follows:

$$S \triangleq \{\underbrace{\text{Exists non-matching}}_{S_0}, \underbrace{\text{All accept}}_{S_1}\} = \{S_0, S_1\}. \quad (1)$$

The event S says that there exists a pair of devices from $\mathcal{G}$ that does not have matching conversations and at *the same time* all the verifications in the GAP protocol (Figure 2) are successful. In other words, there exists a sensor device $i \in \mathcal{G}$ that has accepted a potentially altered message as an authentic one. Therefore, we can define the probability of a successful attack on the GAP protocol as $\mathbf{P}[S]$.

Let us introduce some additional notation. We denote with $view_i$, $i \in \mathcal{G}$ the ordered set comprising all commitments received by the device i, including its own c_i. More precisely, $view_i = \{\widehat{c}_{ID_1 i}, \widehat{c}_{ID_2 i}, \ldots, \widehat{c}_{ID_{i-1} i}, c_i, \widehat{c}_{ID_{i+1} i}, \ldots, \widehat{c}_{ID_{M_i} i}\}$, where $(M_i - 1)$ is the number of commitments received by the device i. To avoid somewhat cumbersome notation, we drop the ID from each index so that finally we have $view_i = \{\widehat{c}_{1i}, \widehat{c}_{2i}, \ldots, \widehat{c}_{i-1 i}, c_i, \widehat{c}_{i+1 i}, \ldots, \widehat{c}_{M_i i}\}$. Please note that the set $view_i$ is ordered with respect to the sender identities. The following fact follows directly from the Phase IV of the GAP protocol.

Fact 3: In a successful attack, the number of commitments received by the coordinator k must be $M - 1$, implying, $|view_k| = |\mathcal{G}_k| = M$.

We next state the following useful result (we omit a straightforward proof for the lack of space).

Lemma 1: If we have non-matching conversation(s) and all the devices from $\mathcal{G}$ "Accept" then either $\exists i, j \in \mathcal{G}$ such that $view_i \neq view_j$ *or* otherwise all the devices "Accept" with the negligible probability ϵ_0.

We continue our proof by introducing another event denoted A:

$$A \triangleq \{\exists (i,j) \in \mathcal{G} \text{ s.t. } view_i \neq view_j\}\ . \quad (2)$$

Then we can bound the probability of a successful attack ($\mathbf{P}[S]$) as follows:

$$\begin{aligned} \mathbf{P}[S] &= \mathbf{P}[S|A] \cdot \mathbf{P}[A] + \mathbf{P}\left[S|\overline{A}\right] \cdot \mathbf{P}\left[\overline{A}\right] \\ &\leq \mathbf{P}[S|A] + \mathbf{P}\left[S|\overline{A}\right] \\ &\overset{(1)}{\leq} \mathbf{P}[S|A] + \epsilon_0 \\ &\overset{(2)}{=} \mathbf{P}[S_1|A] + \epsilon_0 \end{aligned} \quad (3)$$

where (1) follows from Lemma 1 and (2) from the fact that the event A implies that we will have for sure non-matching conversation(s). From the definition of event S_1 and by applying the probability product rule we obtain the following bounds on $\mathbf{P}[S_1|A]$, $\forall i \in \mathcal{G} \backslash \{ID_k\}$, k being the coordinator device:

$$\mathbf{P}[S_1|A] \leq \mathbf{P}\left[\mathsf{GAS}_k = \mathsf{GAS}_i | A, S_{1-(\mathsf{GAS}_k = \mathsf{GAS}_i)}\right]\ , \quad (4)$$

where $S_{1-(\mathsf{GAS}_k = \mathsf{GAS}_i)}$ denotes that all verifications in the GAP protocol, other than $\mathsf{GAS}_k = \mathsf{GAS}_i$, are successful.

Fact 4: If $\exists i, j \in \mathcal{G}$ s.t. $view_i \neq view_j$, then the following holds for the coordinator $k \in \mathcal{G}$: $view_k \neq view_i$ or/and $view_k \neq view_j$.

Then from Fact 4 we know that if the event A has occurred, then there exists $i \in \mathcal{G}$ such that $view_i \neq view_k$. Let i denote such a device. We have two possibilities for devices $(i,k) \in \mathcal{G}$, either $M_i = M_k$ or $M_i \neq M_k$, that is, $M_i = M$ or $M_i \neq M$ (from Fact 3 $M_k = M$). From this and the bounds in equations (3) and (4) one can easily establish (using the law of total probability) the following bound on the probability of a successful attack $\mathbf{P}[S]$:

$$\mathbf{P}[S] \leq \max \left\{ \begin{array}{l} \mathbf{P}\left[\mathsf{GAS}_k = \mathsf{GAS}_i | A, S_{1-ki}, (M_i = M)\right] \\ \mathbf{P}\left[\mathsf{GAS}_k = \mathsf{GAS}_i | A, S_{1-ki}, (M_i \neq M)\right] \end{array} \right. , \quad (5)$$

where S_{1-ki} is a shorthand notation for $S_{1-(\mathsf{GAS}_k = \mathsf{GAS}_i)}$.

What remains to be shown is that both probabilities on the right in inequality (5), are bounded above by $2^{-\ell}$ plus some negligible probability. For simplicity and due to page limitations, we focus only on the case $(M_i \neq M)$.

Condition $M_i \neq M$**.** By definition (Section 3), $\mathsf{GAS}_i = N_i \oplus \widehat{N}_{-i}$, where $\widehat{N}_{-i} \triangleq \oplus_{j \in G_i \backslash \{i\}} \widehat{N}_{ji}$. Similarly, $\mathsf{GAS}_k = N_k \oplus \widehat{N}_{-k}$, with $\widehat{N}_{-k} \triangleq \oplus_{j \in G_k \backslash \{k\}} \widehat{N}_{jk}$. Note that we must have $|G_i| = M_i$ and $|G_k| = M$, because by assumption all verifications other than $\mathsf{GAS}_k = \mathsf{GAS}_i$ are successful. In other words, the number of commitments received in Phase II of GAS has to match the number of IDs received in Phase I, otherwise the protocol is aborted by i and/or k before reaching the final phase of GAS. Now we can write the following:

$$\mathbf{P}\left[\mathsf{GAS}_k = \mathsf{GAS}_i | A, S_{1-ki}, (M_i \neq M)\right] \quad (6)$$

$$= \mathbf{P}\left[N_k = \widehat{N}_{-k} \oplus N_i \oplus \widehat{N}_{-i} | A, S_{1-ki}, (M_i \neq M)\right] \quad (7)$$

$$= \mathbf{P}\left[N_i = \widehat{N}_{-i} \oplus N_k \oplus \widehat{N}_{-k} | A, S_{1-ki}, (M_i \neq M)\right]\ . \quad (8)$$

In order to show that the probability (6) is bounded by $2^{-\ell}$ (plus a negligible probability), we will next show

that N_k is essentially independent of $\widehat{N}_{-k}$ and N_i, and $\widehat{N}_{-i}$, or otherwise N_i is independent of $\widehat{N}_{-i}$ and N_k, and $\widehat{N}_{-k}$.

Independence (N_i, N_k)**.** By the GAP protocol, devices i and k generate N_i and N_k independently of each other.

Independence (N_k, N_{-k}) **and** (N_i, N_{-i})**.** Note that the GAP protocol induces a total order on the set of exchanged messages (see Section 3). The fact that we condition probability (6) on the event S_{1-ki} implies that all verifications other than $(\mathsf{GAS}_k = \mathsf{GAS}_i)$ are successful. From the device i's perspective, a properly ordered exchange of messages with another device $j \in \mathcal{G}\backslash\{i\}$ looks as follows: $\widehat{d}_{ji} \succ d_i \succ \widehat{c}_{ji} \succ c_i$ for $(i < j)$, that is, $d_i \succ \widehat{d}_{ji} \succ c_i \succ \widehat{c}_{ji}$ for $(i > j)$, where the binary operator $\succ$ implies that a variable on the left side succeeds in time the variable on the right side.

We claim that an adversary does not benefit from seeing decommitment d_i when trying to construct $\widehat{c}_{ji}$. Indeed, it follows from S_{1-ki} that at the moment at which i revealed d_i, the device i must already have received all commitments (including a candidate for $\widehat{c}_{ji}$) in the proper order. Any commitment that succeeds d_i contradicts to S_{1-ki}.

Therefore, we ask ourselves: Can the adversary generate $\widehat{N}_{ji}$ (as a part of $\widehat{c}_{ji}$) such that it is related to N_i (from c_i) when only the commitments are available? We consider the scenario where $i < j$ (similar analysis applies to $j < i$). Having seen c_i, the adversary has only four options: **(i)** set $\widehat{c}_{ji} = c_i$, **(ii)** try generate a related commitment such that the covariance $Cov(\widehat{N}_{ji}, N_i) > 0$, **(iii)** break the hiding property of $commit(\cdot)$, and of course **(iv)** try to guess N_i.

In case **(i)**, the adversary has to make sure that the ID_i that appears in c_i (see Section 3) is changed into some different value, otherwise i aborts the protocol (i.e., S_{1-ki} has not occurred). Because $\widehat{c}_{ji} = c_i$, this can be done only by altering the corresponding d_i to obtain $\widehat{d}_{ji} \neq d_i$. From S_{1-ki} we know that $\widehat{d}_{ji} \neq d_i$ opens c_i, implying that the commitment scheme is broken. This can happen only with a negligible probability ϵ_c. Case **(ii)**) implies a successful attack on the non-malleable commitment scheme, which can happen with probability at most ϵ_c. In case **(iii)** the adversary learns the value of N_i from c_i only with the negligible probability ϵ_c, thanks to the (computationally) hiding property of the used commitment scheme. Finally, in case **(iv)**, the probability of success is clearly $2^{-\ell}$, ℓ being the length of N_i.

By summing up all the probabilities we conclude that the adversary can relate an arbitrary $\widehat{N}_{ji}$ (from $\widehat{N}_{-i}$) to N_i with the probability that is at most $2^{-\ell} + 3 \cdot \epsilon_c$.

Independence (N_k, N_{-i}) **and** (N_i, N_{-k})**.** As before, S_{1-ki} implies that all verifications other than $(\mathsf{GAS}_k = \mathsf{GAS}_i)$ are successful. So both devices k and i see well ordered messages. For device k, a proper exchange of messages with any two devices $j, m \in \mathcal{G}_k$ such that $m < k < j$ is as follows: $\widehat{d}_{jk} \succ d_k \succ \widehat{d}_{mk} \succ \widehat{c}_{jk} \succ c_k \succ \widehat{c}_{mk}$. Similarly, for device i and any two devices $j, m \in \mathcal{G}_i$ such that $m < i < j$ we have $\widehat{d}_{ji} \succ d_i \succ \widehat{d}_{mi} \succ \widehat{c}_{ji} \succ c_i \succ \widehat{c}_{mi}$. We ask ourselves if an adversary can relate $\widehat{N}_{ji}$ and/or $\widehat{N}_{mi}$ ($\widehat{N}_{jk}$ and/or $\widehat{N}_{mk}$) to N_k from c_k (N_i from c_i).

As before, we claim that the adversary does not benefit from seeing any decommitments. Let us consider the moment when the first decommitment is sent (revealed). Let this be d_k (similar analysis applies to any decommitment). At this moment, the adversary learns N_k and can adjust accordingly $\widehat{N}_{ji}$ such that $Cov(\widehat{N}_{ji}, N_k) > 0$. Given this, can the adversary generate $\widehat{N}_{mk}$ such that $Cov(\widehat{N}_{mk}, N_i) > 0$ by waiting to receive d_i? We can show that this is not possible. Indeed, from the proper order of messages as seen by the devices k and i above, this attack creates the following two temporal dependencies: $\widehat{c}_{ji} \succ d_k \succ c_k \succ \widehat{c}_{mk}$ and $\widehat{c}_{mk} \succ d_i \succ \widehat{c}_{ji} \succ c_i$. By combining these two temporal chains, we arrive at the following contradiction: $\widehat{c}_{ji} \succ \widehat{c}_{ji}$. Thus, it is not possible to simultaneously relate both $\widehat{N}_{ji}$ to N_k and $\widehat{N}_{mk}$ to N_i. This is valid for any possible combinations of decommitments.

Using similar analysis as in the previous case ("Independence of (N_k, N_{-k})"), we can show that before seeing any decommitment the adversary cannot relate $\widehat{N}_{ji}$ N_k with the probability higher than $2^{-\ell} + 3 \cdot \epsilon_c + \epsilon_h$. The only difference wrt the approach taken in "Independence of (N_k, N_{-k})" is that attacks where the adversary tries to set $\widehat{c}_{ji} = c_k$ are prevented by including $hash(\mathcal{G}_k)$ in c_k and $hash(\mathcal{G}_i)$ in $\widehat{c}_{ji}$ (where $|G_k| \neq |\mathcal{G}_i|$ as $M_i \neq M$); this is the reason for the appearance of the probability (ϵ_h) of finding a collision for $hash(\cdot)$.

From the analysis of independence between N_i, N_k, $\widehat{N}_{-i}$ and $\widehat{N}_{-k}$ and expressions (7) and (8) it readily follows that:

$$\mathbf{P}\left[\mathsf{GAS}_k = \mathsf{GAS}_i | A, S_{1-ki}, (M_i \neq M)\right] \leq 2^{-\ell} + 3 \cdot \epsilon_c + \epsilon_h \ .$$

Finally, from the fact that similar analysis can be carried out for the case $M_i = M$ and expression (5), it easily follows that $\mathbf{P}[S] \leq 2^{-\ell} + \epsilon$, where $\epsilon \triangleq 3 \cdot \epsilon_c + \epsilon_h$.