Blink ’Em All: Scalable, User-Friendly and Secure Initialization of Wireless Sensor Nodes

Nitesh Saxena and Md. Borhan Uddin

Computer Science and Engineering
Polytechnic Institute of New York University
nsaxena@poly.edu, borhan@cis.poly.edu

Abstract. Wireless sensor networks have several useful applications in commercial and defense settings, as well as user-centric personal area networks. To establish secure (point-to-point and/or broadcast) communication channels among the nodes of a wireless sensor network is a fundamental security task. To this end, a plethora of so-called *key pre-distribution* schemes have been proposed in the past, e.g., [25][9][19][8][5]. All these schemes, however, rely on shared secret(s), which are assumed to be pre-loaded onto the sensor nodes, e.g., during the manufacturing process.
In this paper, we consider the problem of user-assisted secure initialization of sensor network necessary to bootstrap key pre-distribution. This is a challenging problem due to the level of user burden involved in initializing multiple (often large number of) sensor nodes and lack of input and output user-interfaces on sensor motes. We propose a novel method for secure sensor node initialization based on a visual out-of-band channel that utilizes minimal output interface in the form of LED(s) already available on most off-the-shelf sensor motes. The proposed method requires only a little extra cost, is efficient and reasonably scalable. Moreover, based on a usability study that we conducted, the method turns out to be quite user-friendly and easy to administer by everyday computer users.

Keywords: Wireless Sensor Networks, Authentication, Key Distribution

1 Introduction

Wireless sensor networks (WSN) have several useful applications in monitoring diverse aspects of the environment. Ready examples include monitoring of: structural/seismic activity, wildlife habitat, air pollution, border crossings, nuclear emission and water quality. In addition to commercial and defense settings, WSNs appeal to a variety of user-centric applications in personal area networks [11, 36, 22]. In some applications, sensor nodes operate in a potentially hostile environments and security measures are needed to inhibit or detect node compromise and/or tampering with inter-node or node-to-sink communication. A large body of literature has been accumulated in the last decade dealing with many aspects of sensor network security, e.g., key management, secure routing and DoS detection [24, 14, 9, 7].

In a WSN environment, the nodes might need to communicate 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 are easy to eavesdrop on and to manipulate, raising the very real threat of the so-called *Man-in-the-Middle* (MiTM) attacker. A fundamental task, therefore, is to secure these communication channels.

Key Pre-Distribution and the Underlying Assumption: A number of so-called "key pre-distribution" techniques to bootstrap secure communication in a WSN have been proposed, e.g., [25, 9, 19, 8, 5]. However, all of them 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. 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 some retailer 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 MiTM attacks. However, sensor nodes might not have any interface other than wireless, 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. To initialize N motes, Bob will have to perform $O(N)$ amount of work. This undermines the scalability of the approach since potentially a large number of sensor nodes might be involved.

Furthermore, we argue that keys can not always be pre-loaded during the manufacturing phase because eventual customers might not trust the manufacturer. Moreover, an application might involve motes produced by multiple manufacturers. A PKI-based solution might be infeasible as it would require a global infrastructure involving many diverse manufacturers.[1]

Secure Initialization Approach: The best possible strategy would be for the network administrator or user of WSN to himself/herself perform the key distribution on-site. Due to lack of hardware interfaces (such as USB interfaces) on sensor nodes and for usability reasons, this key distribution should be performed wirelessly. Prior key pre-distribution schemes assume the existence of some pre-installed secret (such as a point on a bivariate polynomial $f(x, y)$ in [8]) using which the shared keys can be derived. Therefore, the task of key distribution is reduced to establishing a secure channel between the administrator's computer (the sink node) and each sensor node. The resulting secure channels can in turn be used to securely transfer, from the sink to each node, the shared secrets necessary to bootstrap key pre-distribution. Since the administrator might need to initialize a large number of sensor nodes, the process needs to be repeated

[1] The problem that we consider in this paper is very similar to the problem of "wireless 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 [2].

in batches. The larger the number of sensor nodes in each batch, the more *scalable* is the secure initialization method.

Out-of-Band Channels: In quest of a scalable sensor node initialization method, we consider out-of-band (OOB) channels. The OOB (audio, visual or tactile) channels have recently been utilized in the context of secure device pairing application [2, 21, 12, 29], used to establish shared keys between two previously un-associated devices (we review these methods and their applicability to sensor node initialization in the following section). Unlike the wireless communication channel, the OOB channels are both perceivable and manageable by the human user(s) operating the devices, and thus can be used to authenticate information exchanged over the wireless channel. Unlike the wireless channel, the attacker can not remain undetected if it interferes with the OOB channel, although it can still eavesdrop upon it.

Our Contributions: We develop a scalable sensor node initialization method based on a visual OOB channel. Our system builds on an existing protocol of Saxena et al. [29]. However, we make two important extensions to realize the proposed system. First, we develop a new visual channel consisting of simultaneously blinking LEDs[2] as transmitters on sensor nodes and a video camera on the administrator's computer (the sink). This enables efficient transmission of OOB data from sensor motes to the sink with little involvement from the administrator. Second, we design a very intuitive yet effective interface on the sink that allows the administrator to easily discard any potential "attacked" sensor nodes.

Our experiments show that with an inexpensive web cam connected to a laptop or desktop computer, we are efficiently able to use the above visual channel to securely initialize 16 sensor nodes per batch. To evaluate our proposal at the "usability layer," we pursue a thorough and systematic usability study. The results of our study show that our system is both user-friendly as well as robust to errors (human or otherwise).

Organization: In the following section, we review the prior related work. Next, we describe the security model and summarize the relevant protocol we use. This is followed by the description of the design and implementation of our scheme. Finally the results of our experimental testing are presented and discussed.

2 Related Work

The problem of secure sensor node initialization has been considered only recently. Most closely related to our proposal is the sensor network initialization method, called "Message-In-a-Bottle" (MiB) by Kuo et al. [6]. In MiB, the key distribution takes place inside a Faraday Cage, which is used to shield communication from eavesdropping and outside interference. MiB can support key distribution onto multiple sensor nodes[3] and from the administrator's perspective, it is quite user-friendly. However, it has some drawbacks. The first problem is the need to obtain and carry around a specialized piece

[2] Most commercially available sensor motes possess multiple (typically three) LEDs. (For example, refer to Mica2 specifications [1].)

[3] Although it is not clear from the experiments presented in [6], at most how many motes can be initialized per batch.

of equipment – a Faraday Cage. As illustrated in [6], building a truly secure Faraday Cage might be a challenge. The cost and the physical size of the Cage can also be problematic. In other words, only a very few sensor motes could be supported in each batch with a reasonably priced and reasonably sized cage. The second drawback with MiB is that if the initialization process fails for only one sensor node or if there is an error (e.g., if the cage was not properly closed), the entire batch of sensor nodes needs to be re-initialized and re-keyed from scratch. Third, a batch of sensor motes must consist of homogeneous sensor motes with similar weights (the weight is used to calculate the number of motes inside the Cage [6]). Fourth, two additional motes (called "keying device" and "keying beacon") that possess physical interfaces, such as USB connectors, are needed along with the base station and un-initialized nodes. These increase both the cost and the complexity of the system.

The method we propose in this paper can be viewed as an alternative to MiB; the former provides (authenticated key exchange) protocol level security whereas the latter offers physical layer security. Our method also addresses aforementioned drawbacks with MiB (we will discuss this in the final section of the paper). As opposed to MiB [6], our proposal is based on public-key cryptography. We note, however, that most commercial sensor motes are efficiently able to perform public key cryptography [20]. Elliptic-Curve Cryptography has particularly been shown to be very promising on sensor motes [18].

Prior to the MiB method of [6], following schemes were proposed. However, these schemes were aimed at associating only two sensor nodes at a time and not multiple nodes, which is the focus of our paper. The "Shake-them-up" [4] scheme suggests a simple manual technique for pairing two sensor motes that involves shaking and twirling them in very close proximity to each other, in order to prevent eavesdropping. While being shaken, two sensor motes exchange packets and agree on a key one bit at a time, relying on the adversary's inability to determine the sending sensor node. However, it turns out that the sender can be identified using radio fingerprinting [27] and the security of this scheme is uncertain.

Other two related schemes are: "Smart-Its Friends" [13] and "Are You with Me?" [17]. Both use human-controlled movement to establish a secret key between two devices. In addition to having the same drawbacks as "Shake-Them-Up", these schemes would require an accelerometer on each sensor mote to measure movement. Most sensor motes are not equipped with accelerometers, however.

The initialization method that we propose in this paper is similar to the device pairing schemes that use an OOB channel. Thus, we also review most relevant device pairing methods and argue whether or not they can be extended for the application of scalable sensor node initialization. In their seminal work, Stajano and Anderson [35] proposed to establish a shared secret between two devices using a link created through a physical contact (such as an electric cable). As pointed out previously, this approach requires interfaces not available on most sensor motes. Moreover, the approach would be unscalable.

Balfanz, et al. [2] extended the above approach through the use of infrared as an OOB channel – the devices exchange their public keys over the wireless channel followed by exchanging (at least 80-bits long) hashes of their respective public keys over infrared. Most sensor motes do not possess infrared transmitters. Moreover, infrared is not easily perceptible by humans.

Based on the protocol of Balfanz et al. [2], McCune et al. proposed the "Seeing-is-Believing" (SiB) scheme [21]. SiB involves establishing two unidirectional visual OOB channels – one device encodes the data into a two-dimensional barcode and the other device reads it using a photo camera. To apply SiB for sensor node initialization, one would need to affix a static barcode (during the manufacturing phase) on each mote, which can be captured by a camera on the sink node. However, this will only provide unidirectional authentication, since the sensor motes can not afford to have a camera each. Note that it will also not be possible to manually input on each sensor mote the hash of the public key of the sink, since most motes do not possess keypads and even if they do, this will not scale.

Saxena et al. [29] proposed a new scheme based on visual OOB channel. The scheme uses one of the protocols based on Short Authenticated Strings (SAS) [23], [16], and is aimed at pairing two devices (such as a cell phone and an access point), only one of which has a relevant receiver (such as a camera). The protocol is depicted in Figure 1 and as we will see in the next section, this is the protocol that we utilize in our proposal. In this paper, we extend the above scheme to a "many-to-one" setting applicable to key distribution in sensor networks. Basically, the novel OOB channel that we build consists of multiple devices blinking their SAS data simultaneously, which is captured using a camera connected to the sink. In addition, we design an intuitive user interface on the sink that facilitates human users to clearly discard any potential "attacked" sensor nodes.

Recently, Soriente et al. [34] consider the problem of pairing two devices based on an audio channel. Their scheme can be based on the protocol of [29], with the unidirectional SAS channel consisting of one device encoding its SAS data into audio, and the other device capturing it using a microphone. Extending this scheme to initialize multiple sensor nodes in a scalable manner seems hard as it will be hard to decode simultaneously "beeping" sensor nodes.

There are a variety of other pairing schemes, based on manual comparison/transfer of OOB data: [12, 37] can not be used on motes as they require displays; [33, 26] are applicable on sensor motes but would not scale well due to their manual nature.

3 Communication and Security Model, and Protocol

Model: The protocol that we utilize in our initialization method is based upon the following communication and adversarial model [38]. The devices being paired are connected via two types of channels: (1) a short-range, high-bandwidth bidirectional wireless channel, and (2) auxiliary low-bandwidth physical OOB channel(s). Based on device types, the OOB channel(s) can be device-to-device (d2d), device-to-human (d2h) and/or human-to-device (h2d). An adversary attacking the pairing protocol is assumed to have full control on the wireless channel, namely, it can eavesdrop, delay, drop, replay and modify messages. On the OOB channel, the adversary can eavesdrop on but can not modify messages. In other words, the OOB channel is assumed to be an authenticated channel. The security notion for a pairing protocol in this setting is

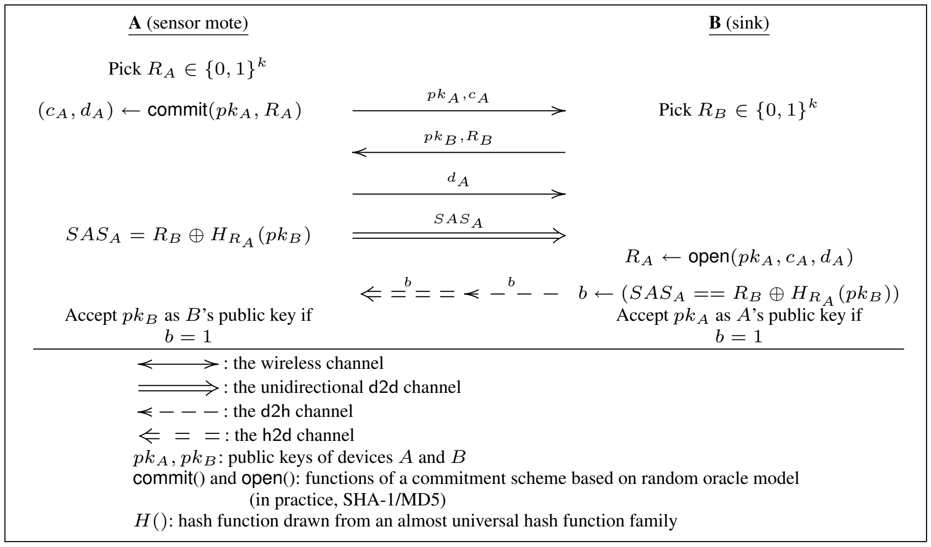

A (sensor mote)
B (sink)
Pick $R_A \in \{0,1\}^k$
$(c_A, d_A) \leftarrow \mathsf{commit}(pk_A, R_A)$
pk_A, c_A
Pick $R_B \in \{0,1\}^k$
pk_B, R_B
d_A
$SAS_A = R_B \oplus H_{R_A}(pk_B)$
SAS_A
$R_A \leftarrow \mathsf{open}(pk_A, c_A, d_A)$
b
b
$b \leftarrow (SAS_A == R_B \oplus H_{R_A}(pk_B))$
Accept pk_B as B's public key if $b = 1$
Accept pk_A as A's public key if $b = 1$
: the wireless channel
: the unidirectional d2d channel
: the d2h channel
: the h2d channel
pk_A, pk_B: public keys of devices A and B
$\mathsf{commit}()$ and $\mathsf{open}()$: functions of a commitment scheme based on random oracle model (in practice, SHA-1/MD5)
$H()$: hash function drawn from an almost universal hash function family

Fig. 1. The protocol by Saxena et al. based on the SAS protocol of Pasini-Vaudenay

adopted from the model of authenticated key agreement due to Canneti and Krawczyk [3]. In this model, a multi-party setting is considered wherein a number of parties simultaneously run multiple/parallel instances of pairing protocols. In practice, however, it is reasonable to assume only two-parties running only a few serial/parallel instances of the pairing protocol. For example, during authentication for an ATM transaction, there are only two parties, namely the ATM machine and a user, restricted to only three authentication attempts. The security model does not consider denial-of-service (DoS) attacks. Note that on wireless channels, explicit attempts to prevent DoS attacks might not be useful because an adversary can simply launch an attack by jamming the wireless signal.

In a communication setting involving two users restricted to running three instances of the protocol, the SAS protocol of [29] [30] need to transmit only k $(= 15)$ bits of data over the OOB channels. As long as the cryptographic primitives used in the protocols are secure, an adversary attacking these protocols can not win with a probability significantly higher than 2^{-k} $(= 2^{-15})$. This gives us security equivalent to the security provided by 5-digit PIN-based ATM authentication.

Protocol: The protocol that we utilize [29][30] is depicted in Figure 1 (we base the protocol upon the SAS protocol of [23], although it can similarly work with other SAS protocol [16] as well). The protocol works as follows. Over the wireless channel, devices A (sensor mote) and B (sink) follow the underlying SAS protocol (due to lack of space, we omit describing the protocol steps over the wireless channel and refer the reader to [29]). Then a unidirectional OOB channel is established by device A transmitting the SAS data, over the d2d channel. This is followed by device B comparing the received data with its own copy of the SAS data, and transmitting the resulting bit b of comparison over the 1-bit d2h OOB channel (say, displayed on its screen). Finally, the user reads the transmitted bit b and accordingly indicates the result to device A by transmitting the same bit b over an h2d input channel.

For our application of secure initialization of sensor nodes, we execute the protocol of [29] in a "many-to-one" setting. Basically, the sink runs serial or (preferably) parallel instances of the pairing protocol over the wireless channel with each of the n motes belonging to a batch. The SAS data, however, is transmitted simultaneously from each mote to the sink. Since the SAS data is transmitted simultaneously by each mote, the sink has no efficient way to figure out what SAS value was transmitted by which of the motes it discovered over the wireless channel. Therefore, the sink accepts the key distribution on a particular mote A if the SAS value (derived from information transmitted over the wireless channel) corresponding to A matches with any of the n SAS values received over the SAS channel. Sensor mote A is therefore accepted with a probability at most $n2^{-k}$ instead of 2^{-k} as in the original "one-to-one" setting. Note that in order to achieve the same level of security offered by a 5-digit PIN-based authentication (as mentioned above), the length of the SAS data should now be $15 + log_2(n)$.

The security of our initialization method is equivalent to the security of the underlying SAS protocol, under the assumption that the administrator correctly discards the motes based on the result (bit b corresponding to each mote) indicated by sink.

4 Our Proposal: Secure Initialization using a Visual Channel

In this section, we describe the design and implementation of an efficient, scalable, user friendly and commercially viable method of secure initialization for sensor nodes. The core of our solution relies on the protocol of [29] executed in a many-to-one setting, as mentioned in the previous section. For transmitting the SAS data of all motes simultaneously over the visual channel, the LEDs of sensor motes are used for ON-OFF encoding, and for receiving the data, video frame based image processing is used on the receiver side.

4.1 Set-up of the Mechanism

In our setup (Figure 3), the administrator's computer (the sink) is connected (using a USB interface) with a sensor node having the functionality of a base station. The sink is also connected with a video camera (a web cam). The motes and the sink communicate over the wireless channel. Sensor motes have their on board displays implemented using two types of LEDs – one Sync LED (used for synchronizing the data transmission between the mote and the sink) and at least one Data LED (used for transmitting SAS data). The Data LEDs can be of any color (same or different), but their color(s) should be different from the color of the Sync LED. The blinking LEDs on motes are used to transmit the SAS data, which is captured using the camera on the sink. The sink matches the received SAS data with its own copy of the acquired SAS data for each mote and based on this, learns whether a particular sensor mote "passed" or "failed" during the process. The sink also displays on its screen the result corresponding to each sensor mote. Based on the result indicated, the administrator must remove or turn off the failed motes. In case the sink is also connected with a printer, the screen indicating the result can also be printed, to better assist the administrator.

4.2 Role of the Administrator

The administrator needs to follow the steps shown in Figure 2. On completion of Step 5, the sink makes use of the resulting secure channels between itself and each sensor mote to bootstrap any of the key pre-distribution schemes, e.g., [8].

Step 1. The administrator turns on the sensor motes and places them on a table, one by one.

Step 2. The administrator presses the "Start" button on the sink. This triggers the sink to sense the nearby motes and signal them over the wireless channel to start an instance each of the protocol of Figure 1. Once done with their SAS data computation, the motes show a "Ready" signal to the administrator by lighting up their red LEDs, and the sink shows the message "Focus the Camera on Ready Motes and Press OK".

Step 3. The administrator adjusts the camera accordingly to capture the LEDs of the ready motes and presses the "OK" button on the sink. The sink sends a "Start Transmission" signal over wireless channel to all sensor motes simultaneously to transmit their SAS data. All the motes transmit their SAS data simultaneously and the camera on the sink captures and decodes the data, and shows the result on the screen and/or prints it out.

Step 4. The administrator turns off the failed motes based on the on-screen or printed output. The turning off of a mote is to be implemented in such a manner that it is equivalent to the mote *rejecting* the protocol instance it executed with the sink. If the administrator does not turn off a particular mote, within an (experimentally determined) time period Δ, by default, the protocol instance will be accepted by the mote. (The default acceptance mechanism is adopted in order to improve the usability of our method. Under normal circumstances, i.e., when no attacks or errors occur, the administrator does not need to turn off any mote.)

Step 5. Steps 1-4 are repeated, batch by batch, until all motes are initialized successfully.

Fig. 2. The Administrator's Role

4.3 Design and Implementation

Our sensor node initialization method requires three phases: (1) the device discovery phase, whereby the sink discovers each mote (over the wireless channel)[4], (2) protocol execution phase, whereby the first three rounds of the SAS protocol of Figure 1 are executed between the sink and each mote, and (3) the SAS data transmission, whereby the sensor motes simultaneously transmit their SAS data, the sink captures them, matches each of them with the local copies and accordingly indicates to the administrator to discard any failed motes.

[4] The sink as well as the motes need to know the actual number n of motes being initialized in one batch, since the length k of random nonces R_A and R_B and of SAS_A in the protocol of Figure 1, should ideally be equal to $15 + log_2(n)$ (as discussed in Section 3). However, an adversary might influence the value of n the sink and the motes determine by sensing over the wireless channel. Therefore, one can hard-code the value of k on the motes and on the sink, based on the expected maximum number of motes to be initialized in a batch. For example, one can safely set k to be 20, if it is expected that at most only 32 motes will be initialized in a batch.

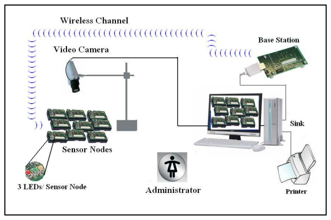


Fig. 3. The Overall Set-up of Mechanism

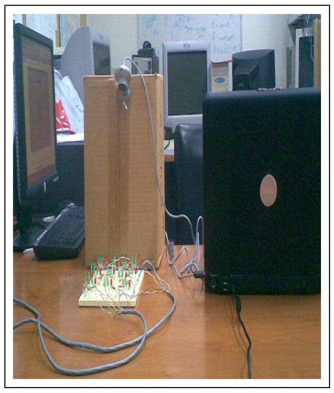


Fig. 4. Experimental setup: Receiver is Web camera, Transmitters are LEDs on Breadboard

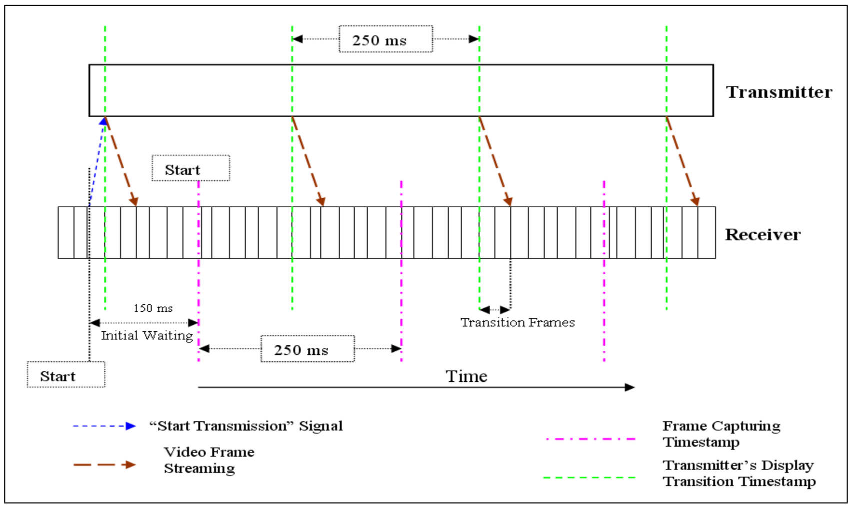


Fig. 5. Synchronization of Transmission (using LEDs) and Reception (on sink) of Data

We were most interested in the third phase as it is an essential element of our proposal. To this end, we have developed an application in Microsoft Visual C# that simulates our sensor node initialization process. The application has two parts – the transmitter simulating the sensor nodes and the receiver simulating the sink; running on two different computers. The transmitter encodes and transmits the SAS data using the display consisting of three blinking LEDs per sensor mote. All motes show the i^{th} bit of their respective SAS data simultaneously. The sink captures the transmitted data as a video stream using its camera, extracts the SAS data for each mote, compares it with its own local copy for the corresponding mote and displays the result on screen and/or prints it out through a printer connected to it. Instead of dealing with real motes[5], we simulated the display of motes using LEDs on a breadboard, integrated with the transmitter through the parallel port of the transmitting computer. It is important to note that our simulated set-up very closely resembles a real system as viewed from usability perspective, which is the primary focus of our evaluation.

Encoding using LEDs: In our simulation, each mote is equipped with one Sync LED(red color LED) for synchronization at the beginning and end of SAS data transmission and two Data LEDs(green color LEDs) for transmitting the SAS data. We simulated the display of a total of 16 sensor motes on a breadboard(Figure 6) each having three LEDs as most commercially available motes; however, our implementation supports an arbitrary number of LEDs (with an arbitrary physical topology) and two distinct but not fixed color LEDs(for differentiating Sync and Data LEDs).

The sync LED (kept “ON” at the beginning and end of SAS data transmission; “OFF”, otherwise) is used to indicate the beginning and end of the SAS data transmission and to detect any synchronization delays, adversarial or otherwise, between the motes and the sink.

The data LEDs are used for SAS data transmission by indicating different bits (‘0’/‘1’) using different states (OFF/ON) of LEDs. If N is the number of Data LEDs, the transmitter can display N bits of SAS data at a time. The states of the sync and data LEDs are kept unchanged for a certain time period (named “hold time”; experimentally determined as 250ms); so that, a stable state (named “BitFrame”) can be easily captured in the video stream of the receiver video camera. After every 250 ms, next N bits of the SAS data are simultaneously shown by each mote in the next frame. This process continues until all bits of SAS data are transmitted. If the last frame does not have N number of SAS bits to show, the beginning required LEDs show the data bits and the remaining are kept OFF.

For discovering the location, color, dimension of LEDs for each mote at the receiver side, two extra frames are needed at the beginning of data transmission – an “All-ON” frame having all LEDs in ON state and an “All-OFF” frame having all LEDs in OFF state. In addition to All-ON and All-OFF frames, another frame is required at the end of SAS data transmission, to detect synchronization delays having the Sync LED in ON state and the data LEDs in OFF state. Therefore, overall a total of three extra frames are required. Thus, for 20-bit SAS data transmission(recall that $[15 + log_2(16)]$-bit long SAS is required for 16 motes) the total number of frames to be transmitted is $\lceil \frac{20}{N} \rceil + 3$,

[5] Since we wanted to deal with a number of motes, a testbed consisting of real motes was not affordable, nor was it necessary.

which yields a total transmission time of $(\lceil\frac{20}{N}\rceil + 3) \times 250$ ms. For transmitting 20-bit SAS data using N=2 data LEDs, there is requirement of a total of 13 frames and thus a total of 3.25 seconds of transmission and capturing time.

Decoding using a Video Camera: For successfully decoding the data transmitted using the LEDs of motes, the receiver video camera must have a frame rate higher than the transmission rate. If frames are not carefully captured from the video stream, there is a likelihood of obtaining the counterfeit frames, which contain the transition state of LEDs.

Resolving the Timing Issue of Frame Capturing: We assume that the transmission delay of "Start Transmission" (ST) signal from the receiver to the transmitter is negligible (5-6 ms) compared to the "hold time" (HT) (of 250 ms) and the receiver video camera also has a delay (about 30-40 ms, since most common cameras have a rate of 30-40 frames per second) of capturing the frame from video stream. Bases on this assumption, the receiver captures the first frame from the video stream after a time, equal to 0.6$\times$ HT (i.e. after 150 ms), termed as "initial waiting" (IW), after sending the signal. The sink pre-calculates capturing (saving frames into memory from video stream buffer) timestamps for all frames by adding the IW + (HT (250ms) $\times$"frame_index"), with the timestamp of sending of the ST signal. The frames are captured into memory at the corresponding timestamps. Figure 5 depicts the synchronization of transmission and reception of SAS data. In this figure each small rectangle on the receiving window denotes a video frame of video stream and brown arrow marked with "Video Frame Streaming" denotes the propagation of transmitted signal to streamed frame in the video stream, which implies that there is some propagation delay of an input transition from transmitter's side to the receiver's video stream.

Detection of LEDs and Retrieval of SAS data: The frames are processed after the completion of capturing of all required frames. Our LED location and dimension detection algorithm is simple yet fast, robust and efficient, unlike existing object/face detection algorithms [28, 31, 39]. The algorithm detects the position and dimension of LEDs deterministically. It is able to detect any shape/geometry of LEDs unlike [39] and does not require any prior training unlike [28, 31]. The algorithm uses the color threshold adjustment technique like [40] to detect the position and dimension of LEDs.

The maximal differences of RGB values, $max(dR, dG, dB)$ (denoted as μ), of each pixel of All-OFF and All-ON frames are measured and kept in memory. Using a threshold value for μ, bit-strings are built for each row of pixels. For example, if μ exceeds a certain threshold, the corresponding bit in the string becomes '1', otherwise it becomes a '0'.

Each bit-string is matched against a regular expression for consecutive 1s. For each matching bit-string, its center is calculated and its safeness and centeredness as an LED center is checked by matching against the already explored LEDs and exploring only the nearby pixels of this center in the frame. If its safeness and centeredness is proved, it is accepted as an LED and its coordinates are included in the explored list of LEDs. This process continues up to a number of times by adjusting the threshold value of μ and constructing the new bit strings until all LEDs are detected. In Figure 7, we show an example of detection of LEDs from the bit-string.

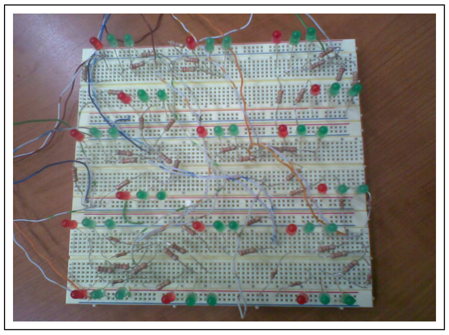
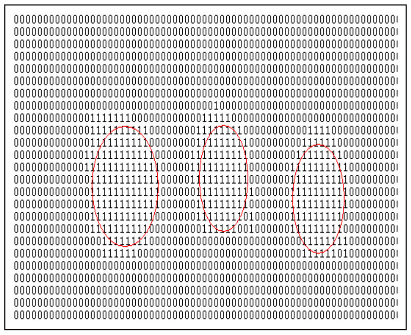


Fig. 6. Transmitter: Breadboard with 48 LEDs Simulating Displays of 16 Motes



```
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000010000000000000000000000000000000
000000000000011111100000000000001111000000000000000000000000000000
000000000000011111111110000000011111111000000000001111000000000000
000000000000011111111111000000011111111100000000011111100000000000
000000000000111111111111100000111111111100000000111111100000000000
000000000000111111111111100000111111111100000000111111111000000000
000000000000011111111111100000011111111100000001111111111000000000
000000000000011111111111100000011111111100000001111111111000000000
000000000000011111111111000000011111110100000001111111110000000000
000000000000011111111110000000001111100100000001111111110000000000
000000000000000111111110000000000000000000000000011111110000000000
000000000000000111110000000000000000000000000000011111010000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
```

Fig. 7. Detected LEDs from BitString

After successful discovery of LEDs, the length, width, average RGB values of ON and OFF states of LED area, for each LED are stored in memory for detecting the ON/OFF state of LEDs in subsequent BitFrames. Successfully discovered LEDs are clustered according to a threshold value of proximity among themselves, for identifying the displays of different sensor motes.

After successful detection of all motes, the data LEDs of each mote are sorted according to the left-to-right and top-to-bottom ordering of coordinates. Now SAS data for each mote is extracted from the BitFrames by comparing the average RGB values of LEDs with previously saved (from All-OFF and All-ON frames) OFF and ON state RGB values of LEDs. For each extracted SAS, the sink matches it with its own computed list of “free” SAS values. If there is a match, the sink marks the corresponding computed SAS as “used” and the mote as “SAS Matched”. If extracted SAS of a mote does not match with any free SAS values, the corresponding mote and all motes having the same SAS are marked as “SAS Mismatched”. Each BitFrame is then examined: the Sync LEDs of all motes should be in the OFF state, except for the last frame, where the Sync LED should be in the ON state and all data LEDs of all motes should be in the OFF state. If this is not the case, it implies that a synchronization error occurred.

If for a mote, both “SAS Matched” and “Sync Matched” are true, the sink accepts the mote as a “passed”; otherwise, it rejects the mote as a “failed” due to mismatch of SAS and/or synchronization errors. The LEDs of a passed mote are marked with a rectangle of green color; and the LEDs of a failed mote are crossed out with red color (Figure 8). Additionally, an automatic printing of the result-screen is done by the printer connected to the sink. By observing the graphical result on screen of the sink and/or the printed result, the administrator discards the failed motes.

5 Experiments and Results

5.1 Experimental Setup

To test our simulator implementing the sensor node initialization method, we used the following set-up. The sink is running on a DELL Vostro 1500 Laptop (1.6 GHz CPU, 2GB RAM, WinXP Pro SP2) connected with a USB Web Camera (Microsoft LifeCam VX6000, up to 30 frames/sec, live video streaming of resolution $640X480$ pixels) and a wireless printer. The webcam can be replaced with any similar camera with a frame rate 30 fps or higher, without any modification to the existing simulator. The camera is set in NON_STOP video capturing mode and frames are taken setting the camera in preview mode. Camera controller is added to the simulator to allow adjusting the focus, tilt and pan of camera as needed.

The transmitting side of the simulator runs on a DELL desktop computer (1.8 GHz CPU, 1 GB RAM, WinXP Pro SP2) connected with LEDs on breadboard (Figure 6) through parallel port (DB25 Connector). The laptop and the desktop computer are connected with our university's wireless connection (54 Mbps). Figure 4 has a snapshot of our set-up.

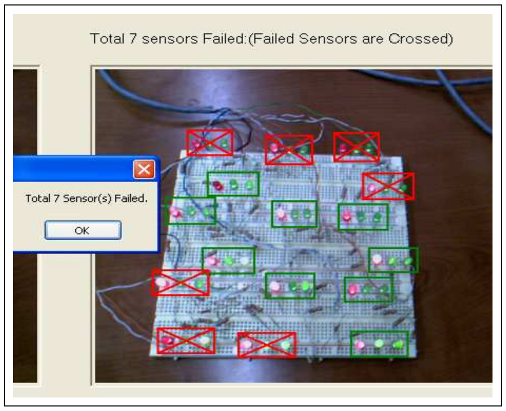
5.2 Usability Testing

In order to test how our method fares with non-expert users, and especially to figure out if the users are easily and correctly able to discard the failed sensor motes based on the result screen (and/or print-out), we performed a usability study.

Testing Framework: For creating an automated testing framework, we extended the transmitter application running on the desktop computer by implementing the usability testing and user feedback collection functionality on it. The sink application running on the laptop was configured to send the result (indicating passed or failed motes) to the desktop application, as soon as it was determined. As there is no interface on breadboard using which the users can turn off the failed mote(s), we simulated the "turning off" mechanism in the desktop application. As soon as the desktop application receives the result from the laptop application, it shows the layout of the mote field (i.e., the breadboard) on screen, associating each sensor mote with a transparent button with the layout of the mote in the background. The users are instructed to transfer the result from the laptop screen to the desktop screen by clicking on the buttons (on the desktop screen) corresponding to the failed motes shown on the laptop screen. After test completion, the desktop application has the functionality of showing the questionnaires to obtain user feedback and logging the data. In our current tests, we did not make use of the printed output.

Test Cases: We created five categories of test cases to evaluate our method against different types of possible attacks and errors. These included (1) matching SAS and no synchronization errors (to simulate normal execution scenarios, where no attacks or faults occur), (2) (single- and multiple-bit) SAS mismatch on a varying number of motes; (3) missing, pre-mature and delayed turning on of the Sync LED (to simulate synchronization errors), (4) both SAS mismatch and synchronization errors, and (5) variable distance (from 0.5 to 2 feet) between the camera and the transmitters. Ten test cases for each category were created. Each user executed a total of five test cases, one each selected randomly from each of the five categories.

A (portion of the) screenshot of the result of execution of one of the test cases is shown in Figure 8.


Total 7 sensors Failed:(Failed Sensors are Crossed)

Total 7 Sensor(s) Failed.

Fig. 8. Result Screen: 7 Failed Motes (marked by "red cross"), 9 Passed Motes (marked by "green rectangle")

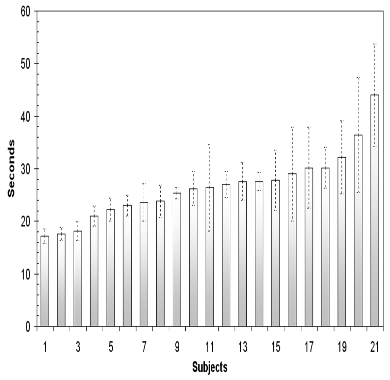


Fig. 9. Average time (per test case execution) taken by 21 subjects with standard error. Subjects are sorted by average time.

Test Participants: We recruited 21 subjects[6] for our usability testing. Subjects were chosen on a first-come first-serve basis from respondents to recruiting posters and email ads. At the end of the tests, the participants were asked to fill out an on screen questionnaire through which we obtained user demographics and their feedback on the method tested.

Recruited subjects were mostly university students, both graduate and undergraduate, with CS and non-CS backgrounds. This resulted in a fairly young (ages between 22-31 [*mean*=25.48, *se*=0.5417]), well-educated participant group. All participants were regular computer users. 19 out of 21 participants reported they have previously used a PC camera (for internet chat). None of the study participants reported any physical impairments that could have interfered with their ability to complete given task. The gender split was: 17 males and 4 females.

Testing Process: Our study was conducted in a graduate student laboratory of our university. Each participant was given a brief overview of our study goals and our experimental set-up. Each participating user was then asked to follow on-screen instructions on the laptop and desktop computer. No training of any sort was given. Basically, the participants played the role of the administrator in the sensor node initialization method, as depicted in Figure 2. Sink output, user interactions throughout the tests and timings were logged automatically by the testing framework.

After completing the deputed test cases in the above manner, the participants were asked to give some qualitative feedback on how easy or hard they found to focus the camera on all LEDs, to read the result of the output screen and about the overall ease/difficulty of the method. Participants demographic information such as age, gender, educational qualification, visual disability, computer and camera experience is also

[6] It is well-known that a usability study performed by 20 participants captures over 98% of usability related problems [10].

collected through this questionnaire. All user data and feedback was logged by the testing framework for future analysis.

Test Results: Each of our 21 subjects executed 5 test cases, leading to a total of 105 test cases. Most of the test cases executed successfully giving expected results. In some cases, however, we observed a few errors, which we categorize and describe below.

- *Camera Adjustment Error:* We configured our usability testing application in such manner that if all the LEDs are not within the camera viewpoint, an error message is shown to the user asking him/her to re-execute. In our tests, 2 users failed to adjust the camera on one occasion each and thus they had to repeat the tests. Therefore, the rate of camera adjustment error equals $\frac{2}{(105+2)} \times 100\% = 1.87\%$ of test cases.
- *Sink Mis-reading Error:* Sometimes the sink is not able to correctly read the SAS string(s) transmitted by one or more sensor nodes. This could happen when the camera is too distant ($>$ 2 feet) from the sensor motes or due to reflection of LED light on the table and other nearby surfaces. In our tests, this type of error occurred for a total of 7 motes, where SAS strings of 1 or 2 motes were mis-read in some 5 test cases. In 105 testcases, the sink dealt with a total of $(105 \times 16) = 1680$ motes on breadboard and out of them 7 motes failed due to sink errors. So, rate of sink mis-reading error equals $\frac{7}{1680} \times 100\% = 0.417\%$. Note that all of these errors were only false positives, i.e., the mistakenly marked a passed mote as a failed one.
- *User Error:* A user error occurs when the user is not able to correctly transfer the result, from the laptop screen to the desktop screen (simulating switching off the failed mote). In our tests, 3 users accidentally clicked, on one occasion each, a passed mote on the desktop screen (this implies that a passed mote was turned off). However, it is important to note that on no occasions did a user miss clicking on a failed mote. In other words, we did get a few false positives but no false negatives whatsoever. Thus, rate of user errors from our tests turned out to be equal to $\frac{3}{1680} \times 100\% = 0.18\%$.

The average time taken by each user (over the 5 test cases), to complete Steps 2 to 4 of Figure 2, is depicted in Figure 9. As we see, the time taken by all of our users to perform a test is less than a minute [*mean*=26.5 seconds, *se*=1.37]. Note that these numbers arise when we assume a fairly conservative setting, one where both normal scenarios and attacks or faults occur with equal likelihood. However, in practice, attacks or faults are less likely. Therefore, considering only the normal test case, we find that that on an average a user only takes 19.18 seconds [*se*=1.11] to complete the whole process.

The results we obtained through the user feedback questionnaire are shown in Table 1. Clearly, most users found the method robust and quite easy to work with. We did not find any notable correlation of the subjects' age, gender and technical expertise with the results obtained for the method, however.

6 Discussion

We proposed a novel method for secure initialization of sensor nodes. Based on the results of our testing with the proposed method, we discuss the following properties.

Easiness	Very Easy	Easy	Medium Difficult	Difficult	Very Difficult	Impossible
Camera Adjustment over LEDs	5	13	3	0	0	0
Detection of Failed Motes	11	10	0	0	0	0
Easiness of Mechanism	7	10	4	0	0	0

Table 1. User Feedback (numbers denote the number of users)

6.1 Efficiency

Using N Data LEDs and one Sync LED per sensor node, the transmission requires $[\lceil \frac{20}{N} \rceil + 3] \times 250$ ms. This is equal to 3.25 sec for N=2 and 20-bit SAS data. Extraction of SAS data from captured frames and displaying the result on screen require less than 3-4 seconds. So, execution time of the method is 7-8 seconds. Overall, as our experiment results show, most users took less than a minute to perform the whole process. Also, as shown in [20], most existing commercial sensor motes (e.g. Mica2) can efficiently execute (within a minute) the public key operations (private and public key generation, and one exponentiation). Note that these operations constitute the dominant costs in the SAS protocol (of Figure 1) that a sensor node executes with the sink. The sink, on the other hand, is assumed to be a computer with a fairly strong computational power and therefore can efficiently execute n parallel protocol instances with each of the sensor nodes.

Based on the above numbers, we recommend setting $\Delta = 2$ minutes, as the time period (to complete Steps 2 to 4 of Figure 2) by which the key initialization will be accepted by each sensor node, by default. As our experiments show, within 2 minutes, a human user can safely complete the initialization process, turning off any (failed) sensor nodes, if necessary.

6.2 Robustness

Our method is quite robust to varying distances between the transmitter and receiver. The distance between the camera and sensor motes on breadboard can be up to 2 feet. The method also works quite well in varying lighting and brightness conditions as it deterministically learns the environment using the first two, All-OFF and All-ON, frames in each session. The method could fail in presence of background noise during transmission and reception of SAS data. Huge variations of lighting conditions during transmission of SAS data which exceed color threshold of LEDs or shaking or displacement of all sensor motes/camera while transmission of SAS data exceeding the dimension threshold of LEDs will also cause failure of the method. However, these will only lead to false positives and not to an attack. Except for the camera adjustment errors (as discussed previous section), all errors occurring with our method are localized i.e, if a single sensor mote fails due to some reason, only that particular sensor node needs to be re-initialized. Note that this is unlike the MiB scheme of [6], where any errors lead to the re-initialization/re-keying of the whole batch of sensor motes. Even when camera adjustment occur in our method, only the SAS data transmission needs be repeated, not the whole initialization process. On the other hand, MiB is less prone to user errors than our method. However, our results indicate that our user errors only lead to false positives and are negligible nevertheless. In our future work, we plan to explore how default rejection (as opposed to our current default acceptance mechanism) would impact the efficiency, usability and scalability of our method. It will clearly improve security.

6.3 Scalability

Our method can be used to initialize multiple sensor nodes per batch. We tested the method with 16 sensor nodes having three LEDs each. Compared to prior work, which only allows for initialization of two motes, this is a significant improvement.[7] By using good quality wide-angle cameras (which will somewhat increase the overall cost of the system), this number can be further improved, we believe. We are currently exploring ways to make our method more scalable. Note that increase in the number of sensor nodes will come at only a slight cost of increase in the length of SAS data. For example, to support 128 sensor nodes, we would need to transmit 22 SAS bits. However, this will make the task of detecting failed motes much harder for the administrator in case the system is under attack by a man-in-the-middle attacker.

6.4 Usability

Via a systematic usability study, we find that our method is quite user friendly. It does not require any expertise or prior training. Little or no acquaintance with the method is enough to administer the process. It is easy to work with and enables safe detection of failed sensor nodes by observing the result on the screen of the sink. Unlike the MiB scheme of [6], the administrator does not have to deal with a specialized and often cumbersome Faraday Cage. Of course, the administrator has to deal with a camera in our method; however, most users are getting more and more familiar with cameras as they become ubiquitous and our usability study (Table 1) shows that most users found the task "camera adjustment to LEDs" to be easy . Moreover, a camera can be used for purposes other than key distribution and is thus not truly specialized. Also note that the sensor motes per batch do not need to be homogeneous. They can have different number, color of LEDs, in any topology/orientation whatsoever (the only requirement being they all possess one RED colored LED to act as the Sync LED). Recall that this is unlike MiB [6], which can only support homogeneous sensor motes with very similar weights. We consider this as an important issue with respect to usability – an administrator might need to initialize a diverse pool of sensor motes and should not need to group them up.

6.5 Power Requirements

From [20], we know that most available commercial motes can do public key crypto operations using only a small amount of power. Now, we show that the SAS data transmission through blinking LEDs also incurs a minimal overhead on motes in terms of power. For 20-bit SAS data transmission, the three LEDs on each mote light-up 13 times (for a period of 250ms), i.e., for a duration of 13×250=3.25seconds. Each LED

[7] Although the MiB [6] method considers multiple sensor nodes, the maximum number of nodes that can be securely initialized per batch is not clear from the experiments and results presented in [6]. We believe this number would be limited by the size of the Faraday Cage used.

has a drop voltage, V= 2.9 Volts (typical range 1.7-3.3 Volts); Current Rating, I= 2.2 mA (typical range 2-3 mA). Therefore, the maximum energy consumption per mote (3 LEDs), E=$3 \times (V \times I \times t)$= $3 \times (2.9 \times 2.2 \times 10^{-3} \times 3.25)$ Volt-A-seconds =0.062205 Joules. As stated in [20], the Energizer No. E91, two AA batteries used in Mica2 motes, have a total energy of $2 \times (1.5 \times 2.850 \times 3600)$=$30780$ Joules. So, our SAS data transmission requires $\frac{0.062205}{30780} \times 100\% = 0.0002\%$ of battery life of Mica2. As shown in [20], public key generation requires 0.816 Joules of energy. Thus, our SAS data transmission is more that 13.11 times better than the public key generation in terms of power consumption

6.6 Simplicity and Economic Viability

The sink needs only a camera and each sensor node requires at least two LEDs (one Sync and one Data) which are very cheap and commonly available. In fact, most existing commercial sensor motes have three LEDs. Our method is quite economic, as opposed to MiB [6] which requires a specialized Faraday Cage and two additional motes having USB interfaces (called "keying device" and "keying beacon") along with a base station and un-initialized nodes.

6.7 Resistance to Malicious Sensor Nodes

Our method offers a natural protection against corrupted or malicious sensor nodes[8]. Our method is based on an authenticated key exchange protocol following the security model of [3]. This model guarantees that an adversary who learns session key(s) corresponding to some corrupted session(s), does not learn any information about the keys corresponding to other uncorrupted sessions. This is unlike MiB [6], where a single corrupted sensor node can compromise keys corresponding to all other sensor nodes[9].

7 Conclusion and Future Work

In this paper, we presented a novel scalable method of secure sensor node initialization. The proposed method offers (authenticated key exchange) protocol level security for key pre-distribution process using visual OOB channel. This is a promising alternative to MiB [6], the only prior work in this area, which offers physical layer security by attenuating and jamming the wireless signals. We believe that achieving physical layer security on insecure wireless channel might be a tough task and require specialized and expensive equipment. Via a thorough and systematic usability study, we showed that our method has several advantages over MiB in terms of scalability, usability, simplicity and economic viability. Our future work includes usability study of the default rejection mechanism as discussed in previous section and improvement of the scalability of the mechanism by using slightly better quality cameras.

[8] A manufacturer could possibly sneak in malicious sensor node(s) along with normal sensor nodes shipped to a customer, as pointed out in [6].

[9] [6] suggests using a software-based attestation technique [32] to prevent this attack.

Acknowledgements

We would like to thank Jonathan Voris for his comments on an earlier version of this paper and CANS'09 anonymous reviewers for their feedback.

References

1. Mica2 specifications. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf`.
2. D. Balfanz, D. Smetters, P. Stewart, and H. C. Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In *Network & Distributed System Security (NDSS)*, 2002.
3. R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *EUROCRYPT*, 2001.
4. C. Castelluccia and P. Mutaf. Shake them up!: a movement-based pairing protocol for cpu-constrained devices. In *International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2005.
5. H. Chan, A. Perrig, and D. X. Song. Random key predistribution schemes for sensor networks. In *IEEE Security & Privacy*, 2003.
6. K. Cynthia, M. Luk, R. Negi, and A. Perrig. Message-in-a-bottle: User-friendly and secure key deployment for sensor nodes. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2007.
7. W. Du, J. Deng, Y. Han, S. Chen, and P. Varshney. A key management scheme for wireless sensor networks using deployment knowledge. In *IEEE INFOCOM'04*, March 2004.
8. W. Du, J. Deng, Y. S. Han, and P. K. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. In *ACM Computer and Communications Security (CCS)*, 2003.
9. L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *ACM Computer and Communications Security (CCS)*, 2002.
10. L. Faulkner. Beyond the five-user assumption: Benefits of increased sample sizes in usability testing. *Behavior Research Methods, Instruments, & Computers*, 35(3):379–383, 2003.
11. G. Giorgetti, G. Manes, J. H. Lewis, S. T. Mastroianni, and S. K. S. Gupta. The personal sensor network: a user-centric monitoring solution. In *BodyNets '07: Proceedings of the ICST 2nd international conference on Body area networks*, 2007.
12. M. T. Goodrich, M. Sirivianos, J. Solis, G. Tsudik, and E. Uzun. Loud and Clear: Human-Verifiable Authentication Based on Audio. In *International Conference on Distributed Computing Systems (ICDCS)*, 2006.
13. 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 Conference on Ubiquitous Computing (Ubicomp)*, Sep 2001.
14. F. Hu and N. Sharma. Security considerations in ad hoc sensor networks. *Ad Hoc Networks*, 3, 2005.
15. C. Karlof, N. Sastry, and D. Wagner. Tinysec: a link layer security architecture for wireless sensor networks. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.
16. S. Laur, N. Asokan, and K. Nyberg. Efficient mutual data authentication based on short authenticated strings. In *International Conference on Cryptology And Network Security (CANS)*, 2006.
17. 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 *International Conference on Pervasive Computing (Pervasive)*, 2004.
18. A. Liu and P. Ning. Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks. In *Information Processing in Sensor Networks (IPSN)*, 2008.

19. D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *ACM Computer and Communications Security (CCS)*, 2003.
20. D. J. Malan, M. Welsh, and M. D. Smith. A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography. In *IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2004.
21. J. M. McCune, A. Perrig, and M. K. Reiter. Seeing-is-believing: Using camera phones for human-verifiable authentication. In *IEEE Security & Privacy*, 2005.
22. A. Milenkovic, C. Otto, and E. Jovanov. Wireless sensor networks for personal health monitoring: Issues and an implementation. *Computer Communications*, 29(13-14):2521–2533, 2006.
23. S. Pasini and S. Vaudenay. SAS-Based Authenticated Key Agreement. In *Public Key Cryptography (PKC)*, 2006.
24. A. Perrig, J. Stankovic, and D. Wagner. Security in wireless sensor networks. *Communications of the ACM*, 47:53–57, 2004.
25. A. Perrig, R. Szewczyk, V. Wen, D. E. Culler, and J. D. Tygar. Spins: security protocols for sensor netowrks. In *ACM Annual International Conference on Mobile Computing and Networking (MOBICOM)*, 2001.
26. R. Prasad and N. Saxena. Efficient device pairing using human-comparable audiovisual patterns. In *Applied Cryptography and Network Security (ACNS)*, 2008.
27. K. B. Rasmussen and S. Capkun. Implications of radio fingerprinting on the security of sensor networks. In *International Conference on Security and Privacy in Communication Networks (SecureComm)*, 2007.
28. H. A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. In *Pattern Analysis and Machine Intelligence(PAMI)*, 1998.
29. N. Saxena, J.-E. Ekberg, K. Kostiainen, and N. Asokan. Secure device pairing based on a visual channel. In *IEEE Security & Privacy, short paper*, 2006.
30. N. Saxena and B. Uddin. Automated Device Pairing for Asymmetric Pairing Scenarios. In *International Conference on Information and Communications Security (ICICS)*, October 2008.
31. H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2000.
32. A. Seshadri, A. Perrig, L. van Doorn, and P. K. Khosla. Swatt: Software-based attestation for embedded devices. In *IEEE Security & Privacy*, 2004.
33. C. Soriente, G. Tsudik, and E. Uzun. BEDA: Button-Enabled Device Association. In *International Workshop on Security for Spontaneous Interaction (IWSSI)*, 2007.
34. C. Soriente, G. Tsudik, and E. Uzun. HAPADEP: Human Asisted Pure Audio Device Pairing. In *International Information Security Conference (ISC)*, Taipei, Taiwan, 2008.
35. F. Stajano and R. J. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Security Protocols Workshop*, 1999.
36. N. Tatbul, M. Buller, R. Hoyt, S. Mullen, and S. Zdonik. Confidence-based data management for personal area sensor networks. In *DMSN '04:1st international workshop on Data management for sensor networks*. ACM, 2004.
37. E. Uzun, K. Karvonen, and N. Asokan. Usability analysis of secure pairing methods. In *Usable Security (USEC)*, 2007.
38. S. Vaudenay. Secure communications over insecure channels based on short authenticated strings. In *CRYPTO*, 2005.
39. P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2001.
40. J. Weszka. A survey of threshold selection techniques. In *Computer Graphics and Image Processing. Vol. 7, 1978.*