Implementation of a User-Friendly, Device Independent, Secure Key Deployment Technique for Sensor Nodes

Kishwar Ahmed[1] Samia Tasnim[2]

Abstract: *One of the most important steps in establishing wireless sensor networks is the deployment of cryptographic keys which, if compromised during initial key setup, could leave the network vulnerable and make it possible for attackers to access transmitted data. Message-In-a-Bottle (MIB), a user-friendly, secure mechanism for key setup on wireless sensor node serves the purpose of implementing cryptographic key on a sensor node; this mechanism relies heavily on metal ware instead of software. This paper focuses on different techniques of implementing cryptographic keys on sensor node, specially the MIB protocol. To the best of our knowledge there has been no such comparative study of these protocols. Additionally, we also focus on the weaknesses of the MIB protocol and propose possible improvements on those faults.*

Keywords: Faraday Cage, Message-In-a-Bottle (MIB), Keying Beacon, Keying Device.

1. INTRODUCTION

The main security challenge for any sensor network is to securely associate devices together. For example, when a device receives data from a sensor, it needs to make sure that the data is received from the sensor it has selected and not from an imposter. Furthermore, integrity, and privacy are often very important too.

The process of securely associating two wireless devices allows two devices, communicating over a short-range radio, to exchange a secret key. This process of exchanging a secret key between sensor nodes is called *pairing protocol*. This key can then be used to authenticate or encrypt subsequent communication. It is important to notice that the key exchanged in a pairing protocol does not need to be authenticated since the identities of sensor nodes do not matter in this context. A user who is

[1] Lecturer, UITS
[2] Software Engineer, M&H Informatics (BD) Ltd.

pairing two devices together only needs assurance that a key has been exchanged between the devices he/she has selected.

A pairing protocol is composed of two separate sub-protocols:

Key exchange sub-protocol: this protocol is run between two wireless devices and results in a secret key shared between the two devices.

Pairing validation sub-protocol: this protocol is executed between the two wireless devices and the user. Its goal is to guarantee to the user that a key has been exchanged between the very two devices he/she actually wished to pair.

Secure key deployment in sensor networks is uniquely challenging. The main focus is to design a protocol which is easy to use despite the demanding requirements of large-scale, secure network deployments:

- No Physical Interfaces
- Secure Key Deployment, Wirelessly
- Key Deployment by Non-Experts
- Batch Deployment for Multiple Nodes

The protocol Message-In-a-Bottle (MIB) supports all the previous requirements and also supports networks with less stringent requirements.

In the next section we give the definition of the problem and the associated assumptions. In Section 3 we discuss the protocol Message-In-a-Bottle (MIB) in detail along with its shortcomings and some possible improvements. Then in Section 4 we discuss related work regarding secure cryptographic key deployment on sensor nodes. Section 5 gives an overall conclusion based on the comparison of the protocols discussed.

2. PROBLEM DEFINITION

The problem discussed in this paper is setting up a shared secret key between a trusted base station and each new uninitialized node when a user gets a set of new sensor nodes. A feasible solution must provide key secrecy, key authenticity, forward secrecy and demonstrative identification. Also, public key cryptography is not a feasible solution as it adds large amounts of software code to the solution.

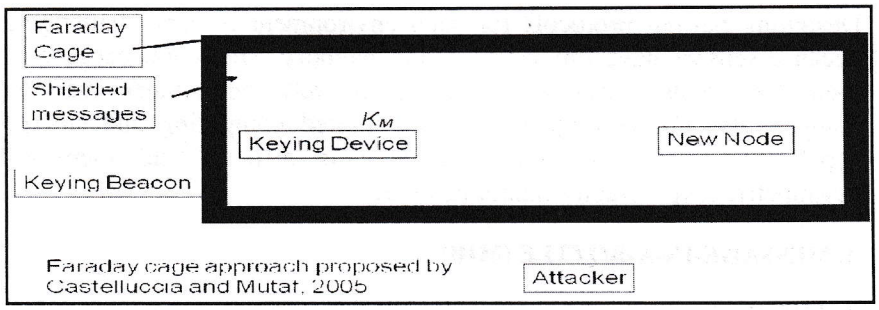

Fig1. MIB Devices and Faraday Cage

Some assumptions are also made on different nodes of the network. It is assumed that the installer is trusted, not an expert but can follow simple directions. Base stations are trusted and can generate keys. Sensor nodes are unmodified hardware and employs secure communication protocol after initial key setup.

The goal of the attacker is to compromise the keys shared by the base station and the sensor nodes. Different protocols assume different attacker model. Message-In-a-Bottle (MIB) assumes Dolev-Yao attacker model. According to this model, an attacker can overhear, intercept, modify, reorder, and send arbitrary messages. It can do these things before, during, and after key deployment. And also, Attacker could be more powerful device with higher antenna gain and faster processor.

Attacker model assumed in protocol *Key Infection* is as follows:

- The attacker does not have physical access to the deployment site during the deployment phase.
- The attacker is able to monitor only a small proportion (α) of the communications of the sensor network during the deployment phase. After key exchange is complete, she is able to monitor all communications.
- The attacker is unable to execute active attacks (such as jamming or flooding) during the deployment phase. After key exchange is complete, she is free to launch any kind of attack.

The motivation of the protocols discussed in this paper (like MIB) is to design a pairing protocol for CPU constrained devices, such as sensors.

Designing pairing protocols for such environment is very challenging because sensors have limited CPU and memory. Moreover, because of their low costs, most of them cannot rely on tamper resistant components. The consequence of the limited computing and storage capabilities is that modular arithmetic is difficult and therefore, asymmetric cryptography cannot be used.

3. MESSAGE-IN-A-BOTTLE (MIB)

A. MIB Devices

Mainly five types of devices participate in MIB: a base station, a keying device, a keying beacon, an uninitialized node, and a user. The goal is to establish a shared secret key between the base station and the new uninitialized node:

The b*ase station* controls the entire network. It also delegates key deployment to two devices: the keying device and the keying beacon. The Key is to be assigned on the *New Node*. It can have three states: uninitialized, initialized, or rejected. The *Keying Device* sends keying information to the new node when the Faraday cage is closed. The *User* of MIB is the person who performs key deployment. The *Keying Beacon* serves three purposes: detect when the Faraday cage is closed; jam the communication channel and inform the user of the outcome of the deployment.

B. Sending Key Wirelessly to New Node

If the keying device sends key to the key node wirelessly then there is a chance for the attacker to eavesdrop on key. So, there is need for some type for isolation. MIB uses Faraday cage (See Figure 1) as an isolation of the keying device and the new node. Faraday cage approach was proposed by Castelluccia and Mutaf, 2005.

But Faraday cage alone is not sufficient. It fails to address the following questions

- How does installer know when to open cage?
- How do nodes know the cage is closed?
- What happens if the Faraday cage is imperfect?
- How does installer know if the node has correct key?

A new device called the keying beacon solves the first problem. Whenever there is key deployment inside the faraday cage this device shows solid blue to mean that key deployment is on progress. When it blinks, it means that key deployment had finished.

Authenticated heartbeats determine whether the cage is closed. Normally a new node and a keying device exchange shielded message. The message exchanged between a keying device and a keying beacon at the initial stage is called authenticated heartbeats. When the cage is closed then no authenticated heartbeat is transmitted and so the new node starts message exchange between the keying devices.

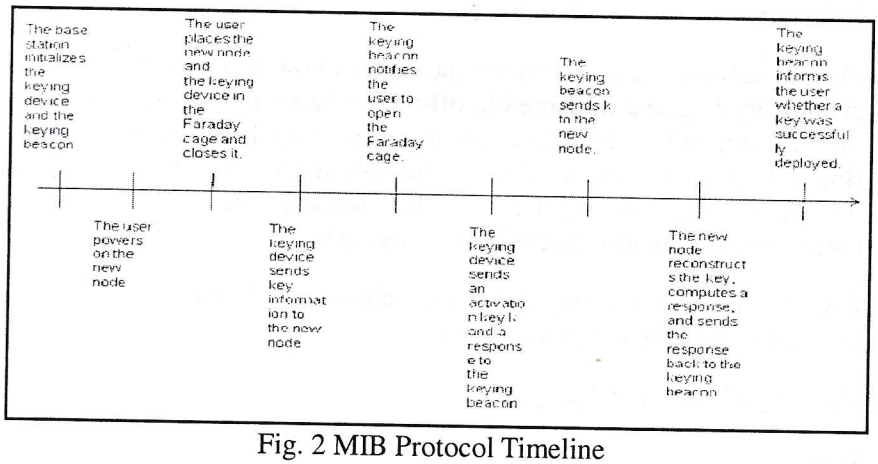

Fig. 2 MIB Protocol Timeline

If the faraday cage leaks then there are two solutions. First, keying beacon eavesdrops. Whenever it hears shielded messages it knows that there is some kind of leak in the cage and so it blinks red light to inform that there is leak in the cage.

Second, keying beacon jams at full power. So, leaked messages are overpowered by the jamming signal. As a result, the leaked messages cannot be heard by eavesdroppers.

To protect shielded messages the following measures are taken in MIB:

- The faraday cage attenuates shielded messages
- Shielded messages sent at minimum power
- The keying beacon jams at full power

- The deployed secret key is a function of all shielded messages.

Finally, to verify that the receiver has got the correct key this protocol uses challenge-response protocol. A challenge is sent to the keying device. The keying device produces response on that challenge. This response is transferred back to keying beacon which sends the response to the new node. The new node then produces response based on that challenge. If this response is same as that produced by the keying device then it is verified, and the new node has been assigned the correct key intended by the keying beacon.

C. Distance of Attacker to Eavesdrop

The attenuation of the cage is the measure of how leaky the cage is. It is denoted by L_{cage} and measured in dBm. The larger this positive numbers the stronger the attenuation. As a result, Attacker cannot overhear shielded messages. The smaller the number the weaker the attenuation. So, the attacker can overhear shielded messages and in this case the keying beacon can also detect leaked messages.

For the attacker to eavesdrop the minimum distance, d_{min} can be calculated from the following equation:

$$RS_e = P_t - 20log_{10}(4\pi d_{min}/\lambda) - L_{cage}$$

Here,

RS_e: Eavesdroppers required radio sensitivity (At least -116 dBm)
P_t : Minimum Transmit power of keying device(-24 dBm)
L_{cage} : Attenuation of cage
L_{cage} = 84 dBm, if no leaking.
L_{cage} = 66 dBm, if cage leaks.
d_{min} : Minimum distance of eavesdropper

To eavesdrop, if the cage does not leak, the attacker needs to be within 2.5 cm and if the cage leaks, attacker needs to be within 19 cm to eavesdrop.

D. Single node key deployment

To deploy cryptographic key on a sensor node using MIB protocol the Figure 2 procedure is followed. The installer places a new node and a

keying device inside the faraday cage and a keying beacon is placed outside the faraday cage. The keying device and the beacon exchange authenticated heartbeats to determine whether the cage is closed. The installer closes the cage and the key is exchanged inside the cage (Shielded messages). The beacon jams at full power so that attacker is unable to eavesdrop in the middle. Beacon notifies installer to open the cage. After the key is deployed on the node, key verification is done using the challenge response protocol. The beacon signals the installer whether the keying was successful.

E. Multiple node key deployment

Placing multiple new nodes in the Faraday cage, it introduces two significant challenges. First, the keying device and the keying beacon need some way to determine when the protocol is completed. In particular, the keying beacon must jam and notify the user to open the Faraday cage at the appropriate times. These times will vary, depending on the number of nodes in the Faraday cage. The keying device also needs to know when all the nodes have received their keys so that it can generate and send the activation keys and validation strings to the keying beacon. Thus, the keying device and keying beacon must know how many new nodes are placed in the Faraday cage. Second, the nodes must be counted without user intervention. Users may miscount the number of nodes – especially as the number of nodes increases. MIB uses a scale to count the number of new nodes. The number of nodes is calculated using the weight of one node. A batch can only contain nodes of the same type. The keying device is then attached to the scale to obtain the reading.

F. Shortcomings of MIB

Although Message-In-a-Bottle is a Wireless, User-friendly, Secure key deployment protocol with low error rate and cost effective; it has a number of shortcomings. A faraday cage cannot be expected to be perfect; even the equipment used in RF testing can only attenuate radio waves [1]. Second, it is unrealistic to assume that there will be no USB (or other) hardware interface. Third, there is an assumption in the protocol that new uninitialized node is trusted. However that may not be the case. Fourth, there are three major arguments against factory-installed keys:

- Customers would need to be confident that attackers would be unable to access or tamper with sensor nodes anywhere along the entire distribution chain.
- Customers would have to trust the manufacturer to manage keys properly (i.e., the manufacturer does not keep a copy of the keys, or nodes initialized for one customer have not been accidentally delivered to another).
- Manufacturers do not want to assume liability for key management.

G. Improvement

One major improvement of Message-In-a-Bottle protocol is the removal of too much dependence on hardware. It uses galvanized pipe as a faraday cage. Hardware is always very expensive and difficult to handle. It is also very unreliable since if it somehow leaks, than there is a good chance of eavesdropping by the attacker. This kind of dependence can be removed by introducing software in place of hardware in the configuration.

To prevent the problem of malicious uninitialized node, software attestation can be used to verify code integrity on all new nodes. Software attestation techniques, such as SWATT [2], allow an external verifier to examine code integrity on an untrusted computing device without hardware extensions. SWATT employs a challenge response based verification function that computes a checksum over the code memory of the untrusted device. The verification function is constructed in such a way that if an attacker modifies the expected code content, either the checksum response would be incorrect, or the execution time of the verification procedure would take longer than expected.

There are some scenarios which are not considered in the problem. For example, what if either the keying device or the keying beacon is the attacker? To solve this problem, we can use a third party device to identify the node as attacker or not. There is also no security concern while providing key to the keying device and keying beacon. We can apply the same protocol after providing the key to the keying device and keying beacon like the one used in deploying key on new node to avoid the problem.

4. RELATED WORKS

There have been many protocols being designed to solve the problem of implementing cryptographic key on sensor node. These protocols can be put into some groups depending on some specific criteria.

A. Physical Interface

1) Resurrecting Duckling. Resurrecting Duckling sets up a secured shared key through the out-of-band channel of physical contact [5, 6]. Because the side channel is assumed to be secured, the key exchanged over this medium is secret and authentic. Unfortunately, this scheme requires a specialized hardware interface for physical contact.

This protocol supports cryptographic key deployment on sensor nodes using direct physical interface between the nodes. This protocol describes secure transient association of a device with multiple serialised owners. It supports all the security properties that might be required, including confidentiality, integrity (and its close relative authenticity) and availability.

A metaphor inspired by biology will help describe the behaviour of a device that properly implements secure transient association. A duckling emerging from its egg will recognise, as its mother, the first moving object it sees that makes a sound, regardless of what it looks like: this phenomenon is called imprinting[5].The devices will recognise as its owner the first entity that sends it a secret key. As soon as this secret key is received, the device is no longer a newborn and will stay faithful to its owner for the rest of its life. If several entities are present at the device's birth, then the first one that sends it a key becomes the owner.

When the device is in the pre-birth state, simply touching it with an electrical contact that transfers the bits of a shared secret constitutes the imprinting. No cryptography is involved, since the secret is transmitted in plaintext, and there is no ambiguity about which two entities are involved in the binding.

*2) Seeing-is-Believing.*In Seeing-is-Believing, an installation device equipped with a camera or a bar code reader, reads a public key on each device that is, encoded as a 2D barcode. Again, since the side channel is assumed to be secured, the key exchanged over this medium is authentic.

Although Seeing-is-Believing does not require special hardware per node, a setup device with specialized hardware is needed. In addition, nodes perform expensive asymmetric cryptographic operations.

With SiB, a mobile phone's integrated camera serves as a visual channel to provide demonstrative identification of the communicating devices to the user while also providing an out-of-band mechanism for exchanging authentic information. By demonstrative identification, it means the property that the user is sure her device is communicating with other device. In SiB, the user identifies other device visually [10]. This serves to strongly authenticated data from the other device since the user knows precisely which devices are communicating. Thus, SiB can be used to bootstrap authentic and secret communication, thereby defeating man-in-the-middle attacks while allowing the use of convenient wireless communication. SiB also captures user intentions in an intuitive way.

The concepts of SiB can be applied in different ways to devices with different capabilities, each equipped with a camera and display, a camera only, a display only, or neither. In some cases, these devices configurations impose some limitations on the strength of the achievable security properties.

SiB has been in use at Carnegie Mellon for several years as part of the Grey Project. SiB has proven to be quite usable for one-on-one exchange of information, such as between two people, or between one person and a device. In addition to the security of the underlying cryptographic primitives, the security of SiB is based on the assumption that an attacker is unable to perform an active attack on the visual channel, and is unable to compromise the mobile device itself.

B. Other Side Channel as Sensors

1) *Talking to Strangers:* Talking to Strangers relies on a location-limited channel, such as audio or infrared, as an out-of-band channel to setup a public key [4]. Like Resurrecting Duckling, this scheme relies on specialized hardware on each device. In addition, Talking to Strangers requires public key cryptography, which is expensive for computationally constrained sensor nodes.

Physical contact is known as location limited channel in this protocol. Location-limited channels have the property that enables human

operators to control which devices are communicating with each other. The notion of location-limited channels was introduced by Stajano and Anderson [5], as a part of their "Resurrecting Duckling" model of interaction in ad-hoc networks. They use secret data exchanged over a contact channel to bootstrap a particular authentication and key exchange protocol ("imprinting" between a "mother" or control device, and a "duckling").

If the participants use the location-limited channel to exchange their public keys as pre-authentication data, it doesn't matter whether an attacker manages to eavesdrop on the exchange. The participants will authenticate each other over the wireless link by proving possession of their corresponding private keys; as the attacker does not know those private keys, he will not be able to impersonate any of the legitimate participants.

2) Shake Them Up: Shake Them Up [3] sets up shared keys between two nodes by requiring the user to hold one in each hand and shake them. These two nodes exchange identical packets, and rely on the fact that the adversary cannot distinguish between messages sent by either device. These two devices, however, could be distinguished using radio fingerprinting [11]. Thus, key secrecy may be violated. Shake Them Up is also not robust against user error. Tired after deploying several nodes, a human technician may deploy nodes without sufficient shaking.

In this protocol security depends on shaking two devices. "Smart-Its Friends" [12], although not related to key agreement nor security, is based on a similar user-device interaction. The authors propose that sensors be equipped with a two axis accelerometer. When a user takes two devices in one hand and shakes them, the devices generate and broadcast similar movement data. If the difference is below a specified threshold, then the two devices recognize each other as friends and a dedicated connection is established between them.

In another related work, "Are you with me?"[8], the authors propose using accelerometers to determine if two devices are carried by the same person. In "Shake Them Up!", the shaking process has a completely different role (and no accelerometers are used in). Devices are shaken/ rotated for randomizing the signal power of the messages received by a potential eavesdropper.

C. Movement of Sensor Nodes

Smart-Its Friends: This is a new pairing protocol that allows two CPU-constrained wireless devices, Alice and Bob, to establish a shared secret at a very low cost. This is the first software pairing scheme that does not rely on expensive public-key cryptography, out-of-band channels (such as a keyboard or a display) or specific hardware, making it inexpensive and suitable for CPU-constrained devices such as sensors.

Table 1. Comparison of different key deployment protocols

	Security		Ease-of-Use		Costs		
	Key Secrecy	Key Authenticity	Demonstrative Identification	Robust to User Error	No per Node extra hardware	No specialized setup Hardware	No public key cryptography
Message In a Bottle	Y	Y	Y	Y	Y	N	Y
Resurrecting Duckling	Y	Y	Y	Y	N	Y	Y
Talking to Strangers	-	Y	Y	Y	N	Y	N
Seeing-is-Believing	-	Y	Y	Y	Y	N	N
On-off-keying	-	N	N	Y	Y	Y	N
Key infection	N	N	N	Y	Y	Y	Y
Shake Them up	N	Y	Y	N	Y	Y	N

In this protocol, Alice can send the secret bit 1 to Bob by broadcasting an (empty) packet with the source field set to Alice. Similarly, Alice can send the secret bit 0 to Bob by broadcasting an (empty) packet with the source field set to Bob. Only Bob can identify the real source of the packet (since it did not send it, the source is Alice), and can recover the secret bit (1 if the source is set to Alice or 0 otherwise). An eavesdropper cannot retrieve the secret bit since it cannot figure out whether the packet was actually sent by Alice or Bob. By randomly generating in such packets Alice and Bob can agree on an n-bit secret key.

This scheme requires that the devices being paired, Alice and Bob, are

shaken during the key exchange protocol. This is to guarantee that an eavesdropper cannot distinguish the packets sent by Alice from those sent by Bob using data from the RSSI (Received Signal Strength Indicator) registers available in commercial wireless cards. The proposed protocol works with off-the-shelf 802.11 wireless cards and is secured against eavesdropping attacks that use power analysis. It requires, however, some firmware changes to protect against attacks which attempt to identify the source of packets from their transmission frequency [12].

5. COMPARISON WITH OTHER PROTOCOLS

Researchers have proposed numerous sensor network key deployment schemes, such as ZigBee, SPINS, LEAP, Transitory Master Key, and random key pre-distribution. Unfortunately, all of these approaches rely on an unspecified secure mechanism to set up the initial secret key in each sensor node.

Some exceptions are Shake Them Up, On-off Keying, and Key Infection. Like MIB, these sensor network key establishment schemes do not rely on pre-shared secrets; hence we will discuss them in detail in this section. We will also compare MIB with out-of-band-based approaches proposed for key setup in ubiquitous computing settings: Resurrecting Duckling, Talking to Strangers, and Seeing-is-Believing. We will discuss each key deployment scheme with respect to several relevant properties: key secrecy, key authenticity, demonstrative identification, robustness to user error, cost effectiveness, and no public key cryptography. To compare cost effectiveness, we discuss two properties: no per-node specialized hardware, and no specialized setup hardware. Table 1 summarizes our comparison.

Resurrecting Duckling sets up a secure shared key through the out-of-band channel of physical contact.

Because the side channel is assumed to be secured, the key exchanged over this medium is secret and authentic. Unfortunately, this scheme requires a specialized hardware interface for physical contact.

Talking to Strangers relies on a location-limited channel, such as audio or infrared, as an out-of-band channel to setup a public key. Like Resurrecting Duckling, this scheme relies on specialized hardware on

each device. In addition, Talking to Strangers requires public key cryptography, which is expensive for computationally constrained sensor nodes.

In Seeing-is-Believing, an installation device equipped with a camera or a bar code reader reads a public key on each device that is encoded as a 2D barcode. Again, since the side channel is assumed to be secure, the key exchanged over this medium is authentic. Although Seeing-is-Believing does not require special hardware per node, a setup device with specialized hardware is needed. In addition, nodes perform expensive asymmetric cryptographic operations.

In On-off Keying, the presence of an RF signal represents a binary '1,' while its absence represents a binary '0' [14, 15]. Assuming that an attacker cannot cancel RF signals, the attacker can only modify authentic messages by changing 0's to 1's – but not the inverse. By carefully selecting the en-coding scheme, On-off Keying ensures that the attacker is unable to modify a packet during transmission.

Key Infection simply sends secret keys in the clear, assuming that an attacker arrives at a later point in time. Designed for simplicity and cost effectiveness, this scheme cannot defend against a determined adversary. If the attacker is actually present during key deployment, she may eaves-drop on the deployed key, violating key secrecy. An attacker may also inject her own keys, violating key authenticity. The lack of user feedback means demonstrative identification is absent.

Shake Them Up sets up shared keys between two nodes by requiring the user to hold one in each hand and shake them. These two nodes exchange identical packets, and rely on the fact that the adversary cannot distinguish between messages sent by either device. These two devices, however, could be distinguished using radio finger-printing. Thus, key secrecy may be violated. Shake Them Up is also not robust against user error. Tired after deploying several nodes, a human technician may deploy nodes without sufficient shaking.

Smart-Its Friends and Are You with Me are two related schemes that use movement to establish a secret key. In addition to sharing the drawbacks of Shake Them Up, these schemes require an accelerometer on each node to measure movement. Because Shake Them Up makes one fewer

assumption than these schemes, smart-Its Friends and Are You with Me are not included in Table 1. As illustrated in Table 1, MIB achieves all but one of the listed properties. Key secrecy and authenticity are attained because MIB ensures that an attacker may not eavesdrop or inject its own key onto the new node. Demonstrative identification is achieved since the user knows that the node in the Faraday cage is the node which receives a key. MIB is robust to user error: any human error (e.g., premature opening of the Faraday cage) results in a failed deployment – rather than key compromise. Furthermore, MIB only requires symmetric cryptographic operations.

MIB requires a special Faraday cage and key deployment nodes with an additional USB interface. However, we argue that MIB is still cost effective because it does not require any specialized hardware per node. This tradeoff is a favorable one: one specialized Faraday cage and two deployment nodes can be used to perform key deployment on many sensor nodes. For large deployments, specialized setup hardware is more economical than additional per-node hardware.

6. CONCLUSION

In this paper, we have first discussed the protocols currently present to deploy cryptographic key on sensor nodes in a wireless sensor network. Positioning key wirelessly on a sensor node is always a difficult problem as there is always chance of an attacker eavesdropping on the network and listening to the message. If the initial key is identified by the attacker then it would be difficult for the sensor nodes to communicate as all the remaining communication will be overheard by the attacker. So, to avoid this problem deploying a key on a node needs some protocols that can do the assignment securely and successfully.

REFERENCES

[1]. Concentric Technology Solutions. RF Shield Box. http://www.rfshieldbox.com/RF Products.htm.

[2]. A. Seshadri, A. Perrig, L. van Doorn, and P. Khosla. Swatt:software-based attestation for embedded devices. In Proceedings of IEEE Symposium on Security and Privacy, May 2004.

[3]. C. Castelluccia and P. Mutaf. Shake them up! A movement-based pairing protocol for cpu-constrained devices. In Proceedings of ACM/UsenixMobisys, 2005.

[4]. D. Balfanz, D. Smetters, P. Stewart, and H. C. Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In Symposium on Network and Distributed Systems Security (NDSS), Feb. 2002.

[5]. F. Stajano. The Resurrecting Duckling - What Next? In Proceedings of Security Protocols Workshop 2000,2000.

[6]. F. Stajano and R. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In Security Protocols, 7th International Workshop. Springer Verlag, 1999.

[7]. H. Chan, R. Anderson, and A. Perrig. Key infection: Smart trust for smart dust. In Proceedings of IEEE International Conference on Network Protocols, May 2004.

[8]. J. Lester, B. Hannaford, and B. Gaetano. Are you with me? - using accelerometers to determine if two devices are carried by the same person. In Proceedings of Pervasive 2004, 2004.

[9]. J. McCune. Reducing the Trusted Computing Base for Applications on Commodity Systems. PhD thesis, School of Electrical and Computer Engineering, Carnegie Mellon University Pittsburgh, PA 15213.

[10]. J. McCune, A. Perrig, and M. K. Reiter. Seeing-Is-Believing: Using camera phones for human-verifiable authentication. In Proceedings of IEEE Symposium on Security and Privacy, May 2005.

[11]. K. B. Rasmussen and S. Capkun. Implications of radio fingerprinting on the security of sensor networks. In Proceedings of IEEE SecureComm, 2007.

[12]. 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 Proceedings of Ubicomp 2001, 2001.

[13]. Winter, D.: Biomechanics and Motor Control of Human Movement (2^{nd} ed.). New York: Wiley (1990).

[14]. M. Cagalj, S. Capkun, and J.-P. Hubaux. Key agreement in peer-to-peer wireless networks. Proceedings of the IEEE (Special Issue on Security and Cryptography), 94(2), 2006.

[15]. M. Cagalj, S. Capkun, R. Rengaswamy, I. Tsigkogiannis, M. Srivastava, and J.-P. Hubaux. Integrity (I) codes: Message integrity protection and authentication over insecure channels. In IEEE Symposium on Security and Privacy, May 2006.