Securing Wireless Sensor Networks: Security Architectures

David Boyle
Department of Electronic and Computer Engineering, University of Limerick, Limerick, Ireland
Email: David.Boyle@ul.ie

Thomas Newe
Department of Electronic and Computer Engineering, University of Limerick, Limerick, Ireland
Email: Thomas.Newe@ul.ie

***Abstract*—Wireless sensor networking remains one of the most exciting and challenging research domains of our time. As technology progresses, so do the capabilities of sensor networks. Limited only by what can be technologically sensed, it is envisaged that wireless sensor networks will play an important part in our daily lives in the foreseeable future. Privy to many types of sensitive information, both sensed and disseminated, there is a critical need for security in a number of applications related to this technology. Resulting from the continuous debate over the most effective means of securing wireless sensor networks, this paper considers a number of the security architectures employed, and proposed, to date, with this goal in sight. They are presented such that the various characteristics of each protocol are easily identifiable to potential network designers, allowing a more informed decision to be made when implementing a security protocol for their intended application. Authentication is the primary focus, as the most malicious attacks on a network are the work of imposters, such as DOS attacks, packet insertion etc. Authentication can be defined as a security mechanism, whereby, the identity of a node in the network can be identified as a valid node of the network. Subsequently, data authenticity can be achieved; once the integrity of the message sender/receiver has been established.**

***Index Terms*—Wireless Sensor Networks, Security, Authentication, Cryptography**

I. INTORDUCTION

This paper is an extended and enhanced version of [1]. Whilst retaining its' original format and content, all sections have been enlarged; incorporating more in-depth analysis of the background, security architectures described and characteristics discussed. There is more technical detail included and discussion regarding the appropriateness and usefulness of the systems, supported by more recent research.

A Wireless Sensor Network (WSN) can be defined as a group of independent nodes, communicating wirelessly over limited frequency and bandwidth [2]. The novelty of WSNs in comparison to traditional sensor networks is that they depend on dense deployment and coordination to execute their tasks successfully. This method of distributed sensing allows for closer placement to the phenomena to be achieved, when the exact location of a particular event is unknown, than is possible using a single sensor [3].

Consider the Crossbow "MICAz" mote [4], currently a typical mote used in WSNs. It consists of a battery, microprocessor (Atmega128), RF transceiver, ADC, 128K bytes Program Flash Memory and 4K bytes EEPROM. It is evident that there are limitations to what can be achieved through networking a number of these motes. Areas such as power management, network discovery, control and routing, collaborative signal and information processing, tasking and querying, and security are all currently under research [5].

Battery powered nodes are a common feature of many WSN applications, where recharging or replacement would not normally be feasible, and so are considered to be disposable. Many methods of powering these devices have been explored, including solar power, but they remain to be seen typically as "one-use" devices [6]. Eventual failure is expected and so maximizing their lifetime and productivity is extremely important. This notion of battery conservation extends to the primitives used to enforce security in WSNs. Security protocols strive to be light-weight, in terms of code size and processing requirements, whilst retaining their usefulness, in order to assist in achieving this goal.

To design a completely secure WSN, security must be integrated into every node of the system. Any component of a network implemented without any security could easily become a point of attack. Resultantly, this dictates that security must pervade every aspect of the design of a wireless sensor network application that would collect or disseminate sensitive information; i.e. requiring a high level of security [7].

Conventional networks require protection against eavesdropping, injection or modification of disseminated data packets, and accordingly, most applications of WSNs require the same protection. Cryptography is the standard method of defense against such attacks [7]. This defense brings with it a number of other trade-offs. Varying levels of cryptographic protection implies a

proportionately varying level of overhead; in the form of increased packet size, code size, processor usage etc. This is the stem of all debate relating to optimal security techniques in WSNs.

Security in WSNs can be defined as the method of protecting a prospective application against all known types of attack. Attacks including denial-of-service (DOS), traffic analysis, multiple identity/node replication, confidentiality and physical tampering are all areas for concern within WSN security architecture design,

It is extremely important to ensure that all known attacks are defended against when designing a security system for a WSN. The success of the application will depend largely upon its reliability and robustness against attack. Consider the applications described in Section I.*A*.

There are many obstacles and constraints involved when designing a security protocol for WSNs. The limited memory, storage and processor capabilities, coupled with stringent power limitations distinguish WSN security architecture design requirements from any other. Harsh environmental operating conditions, the threat of physical compromise and unreliable data transfer also provide challenges for designers.

Intended application areas are intrinsic to what type/level of security is required. Limited only by what can be technologically sensed, the door is open for applications of WSNs in all walks of life. The following subsection briefly describes some application areas of wireless sensor networking.

The remainder of the paper is organized as follows: the Communications Systems involved in WSNs, their importance and implications are considered in Section II. Section III provides more specific analysis on the types of attack to which WSNs are susceptible. Section IV describes the various efforts to design optimal security architectures for WSNs which have been specified/implemented to-date. Sections V describes some of the related work in the field, and Sections VI and VII present a conclusion and some future work.

A. Applications

A current survey of applications of WSNs is provided in [8], in addition to the technologies and standards involved. Two of the most security-oriented applications of WSNs are military and medical solutions.

Consider the military. Secrecy is part of its nature; and data (sensed/disseminated/stored) is required to remain confidential. This is critical to the successful operation of a military application. Enemy tracking and targeting are among the most useful applications of wireless sensor networking in military terms. The most current work can be found in the Defense Advanced Research Projects Agency (DARPA) website [7, 9].

More recently, the idea of "tele-health" or "M-Health" is being embraced. Plug and play wireless sensors have been implemented, forming a body area network (BAN), which is capable of remotely monitoring patients' vital signs and transmitting this information back to the health authorities (via laptop/PDA etc.). Such applications imply that outpatients can be monitored from their homes, freeing space in hospital wards [10]. As physiological patient data is legally required to be kept confidential, the implemented WSN must invoke strong security protocols.

Deployments of wireless sensor networks also exist in areas such as environmental, industrial and commercial monitoring [8]. Again, these have varying security requirements. A universal, scalable security solution would be highly desirable. This could be achieved through the use of a common communications protocol, allowing an application-specific level of protection to be set.

II. COMMUNICATIONS SYSTEMS

The successful operation of wireless sensor networks can be largely attributed to the role played by the communications protocol employed. Networking primitives including architecture, data rates, network size, span, power management and security protocols are all affected, if not dictated, by the communications protocol chosen for the application. Standardization has yet to occur for a communication system optimal for wireless sensor networking. The choices have been considerably narrowed down with the specification of a number of low-power wireless communications protocols; including Bluetooth [11], the IEEE 802.15.4 standard [12] and ZigBee [13], to name but a few. A brief discussion of the methods used to achieve wireless communication between motes using TinyOS is included.

A. Bluetooth

Bluetooth (IEEE 802.15.1) was originally designed to be the industry standard for low power wireless devices. It remains, however, unsuitable for use with many wireless sensor networking applications. This is explicitly as a result of the topology and protocol design. The requirement for all devices in a Bluetooth Piconet to remain synchronized to within a few microseconds is troublesome, despite the channel hopping period of only 600 microseconds which enables the low-latency and high throughput operation of these devices. The expense of entering and exiting a Bluetooth network is also concerning. Using a standard Bluetooth configuration, it can take over 2.4 microseconds to establish a connection and typical Bluetooth radios can consume hundreds of milliwatts whilst monitoring the channel [11]. Because of the shortcomings of Bluetooth in relation to WSNs, compatible security protocols will not be further investigated. A more thorough explanation can be found in [14]. The focus is, therefore, forced upon those that operate in conjunction with the IEEE 802.15.4 standard and the ZigBee specification.

B. IEEE 802.15.4

The IEEE 802.15.4 standard is the IEEE standard for low-rate wireless personal area networks (LR-WPANs). Unlike wireless local area networks (WLANs), connections effected via WPANs involve little or no infrastructure. First released in 2003, this is set to become the standard communications protocol for use in wireless sensor networks. Features allow small, power efficient, inexpensive solutions to be implemented for an expansive range of devices. The main objectives of an LR-WPAN

are ease of installation, reliable data transfer, short-range operation, and extremely low cost and reasonable battery life, whilst maintaining a simple and flexible protocol [12]. More specific details on the types of devices, network topology and layers defined by the standard can be found in [12] and [15]. The standard itself defines the physical (PHY) and MAC layers, component devices and supported network topologies in detail; in addition to other important information.

There are a number of security suites specified in this standard, which will be addressed in the following sub-section. As a result of the large (and continually expanding) number of targeted application areas of this protocol, the processes for key exchange and authentication are not defined by the standard [16]. Because of this, the security architecture described in the standard will be addressed now, and omitted from further comparison later on; however, all of the properties of this standard can be invoked via the ZigBee specification.

1) IEEE 802.15.4 Security

There are a number of security suites that can be implemented under the IEEE 802.15.4 standard. The most basic can be defined as the secured mode or the unsecured mode (i.e. the null security suite has been chosen). ACL mode provides some limited security services, only allowing the receiving of frames from nodes on the devices Access Control List (ACL). A link layer protocol provides the four basic security services. These include access control, message integrity, message confidentiality and replay protection. An application sets its requirements by setting the appropriate parameters in the radio stack. If there are no parameters entered, then, by default, there is no security enabled. Access control (achieved via the ACL) and message integrity ensures that unauthorized parties should be prohibited from participating in the network. Legitimate nodes should be able to detect messages from unauthorized nodes and reject them. Sequential freshness checks are employed to prevent replayed messages from being accepted by the receiver, as the receiver checks the counter, and rejects any message that has the value equal to or less than the previous obtained counter value. To ensure message authentication and integrity, a message authentication code (MAC) is appended to each message sent. The MAC is viewed as a cryptographically secure checksum of the message [17].

Computing the MAC requires senders and receivers to share a secret cryptographic key, and this key is part of the input to the computation. The sender computes the MAC over the packet and includes it with the packet (using the secret key). A receiver sharing the same key re-computes the MAC and compares it with the MAC in the packet. If the two are the same then the receiver accepts the packet, or otherwise rejects it. Message authentication codes must be difficult to forge without a secret key and, resultantly, if an adversary to the network changes a valid message or introduces a phony message, then it would be unable to compute the corresponding MAC, and authorized receivers will reject any of their attempts to damage the network [17].

TABLE I. SECURITY SUITES DEFINED BY IEEE 802.15.4

Name	Description
Null	No Security
AES-CTR	Encryption only, CTR Mode
AES-CBC-MAC-128 AES-CBC-MAC-64 AES-CBC-MAC-32	128 bit MAC 64 bit MAC 32 bit MAC
AES-CCM-128 AES-CCM-64 AES-CCM-32	Encryption & 128 bit MAC Encryption & 64 bit MAC Encryption & 32 bit MAC

The standard defines 8 different security suites (Table I). The security suites can be more broadly classified by their properties. This first of these is the Null suite and provides no security. The next is encryption only (AES-CTR), followed by authentication only (AES-CBC-MAC), and finally encryption and authentication (AES-CCM) [12].

Encryption is performed using the AES encryption algorithm, also known as Rijndael. This is defined in the National Institute of Standards and Technology (NIST) Federal Information Processing Standard (FIPS) Publication 197 [18]. This algorithm has been a US Government standard since May 2002, and is used by their organizations to protect sensitive information. AES-CTR (counter mode of cryptographic operation with AES) means that the CTR mode uses AES as the block cipher; and provides access control, data encryption and optional sequential freshness.

Authentication is done using the cipher block chaining with message authentication code (CBC-MAC), which creates a message integrity code using a block cipher in CBC mode, and computes a MAC over the packet and includes the length of the authenticated data. The code can be computed upon packet reception and can be compared with the one received. The IEEE 802.15.4 standard itself includes a detailed description of this process [12].

AES-CCM is a combination of the encryption and authentication suites detailed above. It has three inputs; the data payload to be encrypted and authenticated, the associated data (header etc.) to be authenticated only, and the nonce to be assigned to the payload and the associated data [15]. Table I illustrates that there are varying MAC lengths to choose from for AES-CBC-MAC and AES-CCM modes of operation (4, 8 or 16 bytes), allowing for some scalability of security depending on application requirements.

C. ZigBee

ZigBee is an industrial consortium, which was designed to build a standard data link communication layer for use in ultra low power wireless communications [13]. The members of this organization came together because they felt that "existing standard technologies were not applicable to ultra-low power application scenarios" [19]. The ZigBee network layer (NWK) is designed to operate just above the PHY and MAC layers specified in the IEEE 802.15.4 standard.

The main responsibilities of the ZigBee NWK layer include the mechanisms used to join and leave a network, apply security to frames and to route frames to their intended destinations. The ZigBee specification also details extra security services, including the processes of key exchange and authentication, in addition to those provided under the IEEE 802.15.4, upon which it is built. These will be further examined in the following sections.

D. Active Messages (TinyOS)

At this juncture it is worth mentioning the packet protocol employed by TinyOS [20], the operating system upon which most of the development of WSNs is carried out. This is an event driven operating system written in network embedded systems C (nesC) [21], an abstraction of the C programming language, and is the platform upon which a large amount of WSN specific security algorithms are designed.

The basic network abstraction of TinyOS is an active message. This is a single-hop, unreliable packet. They can be of variable length, up to a fixed size, and have a destination address and provide synchronous acknowledgements. In order to allow components to be build upon this abstraction, active messages have a type field; essentially a protocol identifier. A detailed description of their operation is available in [22].

III. THREATS TO WSNS

In order to appreciate the challenge of securing a WSN against attack, it is necessary to consider the possible threats to its security. There are a large and increasing number of threats and attacks to which WSNs are susceptible. They can be broadly classified as attacks against the privacy of the network data, denial of service (DOS) attacks, impersonation or replication attacks and physical attacks.

In addition to the types of attack, it is also worth considering that attacks can be launched at any point in the network. This implies that certain attacks may be more effective at different layers of the communications protocol, for example. Table II depicts the various attacks that can be launched at different layers of the communications stack (similar to the IEEE 802.15.4 / ZigBee; based on the OSI model).

DOS attacks take many forms, as can be seen in the table, and are known to be any attack that can undermine a network's capacity to perform its expected functions. In the case of wireless networks, "jamming" the channel with an interrupting signal is an effective attack, as are flooding or collision attacks, for example.

The Sybil attack is considered to be that of a malicious node taking on multiple identities. The node can then launch a number of attacks such as negative reinforcement, or stuffing the ballot box of a voting scheme, for example. This attack is most effective in the higher layers of the communications protocol [23].

The physical security of network nodes is another concern, as it cannot be ensured in certain environments. Nodes are therefore vulnerable to physical harm, or tampering (i.e. reverse engineering).

TABLE II. SENSOR NETWORK LAYER AND ATTACK

Layer	Attack
Physical Layer	DOS – Jamming, Tampering Sybil
Data-link Layer	DOS – Collision, Exhaustion, Unfairness Interrogation Sybil – Data aggregation, Voting
Network Layer	DOS – Neglect & Greed, Homing, Misdirection (Spoofing), Black Holes, Flooding Sybil Wormhole Attack
Transport Layer	DOS – Flooding, De-synchronization

Traffic analysis attacks are forged where the base station is determinable by observation that the majority of packets are being routed to one particular node. If an adversary can compromise the base station then it can render the network useless [24].

Node replication attacks can occur if an adversary can copy the node identification of a network node. In this manner packets could be corrupted, misrouted or deleted, and if this adversary could perform this replication it is possible that cryptographic keys could be disclosed. This could be catastrophic for the network [25].

Network traffic is also susceptible to monitoring and eavesdropping. This should be no cause for concern given a robust security protocol, but monitoring could lead to attacks similar to those previously described. It could also lead to 'wormhole' or 'black hole' attacks [26]. These are routing attacks where an adversary convinces a network node of a shorter, or zero, path to the base station, for example, and can disrupt the network in this manner.

Much work has been carried out attempting to define all of the possible threats to WSNs, but they are sure to evolve as time passes – a lesson learned through the development of the Internet and its associated security mechanisms. A recent survey of the threats to WSNs can be found in [27], for a more accessible description of many of the attacks previously listed.

Considering the many angles of attack, it is necessary to secure every aspect of the WSN to ensure its successful operation. The requirements of a secure sensor network include data confidentiality, integrity, freshness, availability, autonomy (in terms of organization) and authentication. These will be further elaborated upon in the next section.

IV. SECURITY ARCHITECTURES

This section provides details of proposed and implemented security architectures optimal for use with wireless sensor networks. These architectures will be reviewed, contrasted and compared based on their individual characteristics. All are compatible with the communications systems previously described. It is intended that this comparison will also illustrate the

progression of research in the area over the past number of years. As the IEEE 802.15.4 standard has neglected to include an active protocol for implementing its security suites, and the fact that ZigBee can invoke all/any of them, it has been omitted from this section.

The confidentiality and reliability of data, sensed and disseminated, can be vital to the success of many of the applications that WSNs are used for. Data encryption and node authentication are the main defenses against attack. There are numerous methods of encryption and authentication protocols available for implementation in WSNs resulting from the continuous development of the Internet. They do not, however, relate directly to sensor networks, as there are a large number of hardware resource constraints present that do not exist for such Internet intended hardware; i.e. PCs, notebooks, PDAs etc. Wireless sensor networks are, more often that not, considered to be one-off deployments of battery-powered, application specific nodes. This dictates that extensive on-chip processing to execute complex encryption/decryption techniques is not a viable option. Optimizing network lifetime is a major goal of research in the area, and accordingly, the most powerful lightweight solutions are sought, as opposed to top-end Internet based solutions that require greater amounts of processing power, in order to realize this objective.

The debate continues as to which is the best cipher to use, or whether hardware or software encryption is optimal for sensor networks. The most effective way to provide reliable authentication for WSNs is also under major scrutiny by many research groups. Each of these issues will be addressed as they arise in the following security architectures.

In order to achieve a thoroughly secure system, all of the possible attacks must be taken into consideration. These range from eavesdropping on network communication, to denial-of-service attacks and insertion of bogus or malicious packets. A comprehensive review of these attacks can be found in [17].

Whilst allowing for detection, node authentication can also prevent most of the damage that can be done by malicious intruders. Authentication is a mechanism whereby the identity of a node in a network can be identified as a valid member of the network and as such data authenticity can be achieved. This is where the data is appended with a message authentication code (MAC) and can only be viewed by valid nodes capable of decrypting the MAC, through some determinable means. Any messages received from unauthorized network users can be discarded.

There are a number of methods to achieve authentication. These range from device-to-device protocols, where each node authenticates its neighbor's identity, to broadcast protocols, which enable a sender to broadcast critical data and/or commands (in-network reprogramming, for example) to sensor nodes in an authenticated way such that an attacker cannot forge any message from the sender [28]. Traditional broadcast authentication techniques (such as public-key based digital signatures) are not desirable due to the energy constraints on nodes.

A. SPINS

Perrig *et al.* (2002) proposed SPINS, a suite of security protocols optimized for sensor networks [29]. SPINS has two secure building blocks, namely Secure Network Encryption Protocol (SNEP) and μTESLA, which run on top of the TinyOS operating system. SNEP is used to provide confidentiality through encryption and authentication; whilst also providing integrity and freshness. μTESLA is used to provide authentication for broadcasted data.

1) SNEP

At the time, this building block provided a number of unique properties. These include a low overhead (8 bytes per message), kept-state at each end point to remove the need for counter values to be transmitted, and semantic security; a strong security property which prevents eavesdroppers from inferring any of the message content from the cipher text (achieved through incrementing the counter for each message and randomization; the message to be encrypted is preceded by a random bit string, implying no content can be inferred from encrypted messages if an attacker knows plaintext-ciphertext pairs encrypted with the same key). Additionally, data authentication, replay protection and weak freshness are provided.

Under SNEP, communicating nodes share a secret master key, from which they derive independent keys using a pseudorandom function. The processes involved are detailed in [24]. A secure and authenticated message using SNEP is as follows,

$$\mathrm{A} \rightarrow \mathrm{B}: \{D\}_{\{K_{AB},C_A\}}, \mathsf{MAC}(K'_{AB}C_A \| \{D\}_{\{K_{AB},C_A\}}), \quad (1)$$

where A and B are two communicating nodes. D is the data, encrypted with derived independent key K_{AB} and the counter value of A; C_A. The message authentication code is $\mathsf{MAC}(K'_{AB}C_A \| E)$; where K'_{AB} is the derived independent key for the MAC operation, and the encrypted data E is $\{D\}_{\{K_{AB},C_A\}}$. It is worth noting that independent keys are derived for encryption and authentication primitives in keeping with good security design practice.

It is evident that data authentication is achieved through the use of a MAC. The use of the counter value implies that replay protection is invoked, in addition to weak freshness. Overhead is reduced by keeping counter state at each end; removing the need for it to be sent with each message. The exchange process for synchronization is also presented in [29].

In order to achieve strong freshness, a nonce (random number long enough that an exhaustive search of all possible nonces is not feasible) is used by one of the communicating parties. Node A generates nonce N_A at random and sends it to B with a request message R_A. Strong freshness is achieved as B returns the nonce with the response message R_B in an authenticated protocol. This protocol operates as follows:

$$\mathrm{A} \rightarrow \mathrm{B}: N_A, R_A, \quad (2)$$

$$B \rightarrow A: \{R_B\}_{\{K_{BA},C_B\}}, \mathrm{MAC}(K'_{BA}, N_A \| C_B \| \{R_B\}_{\{K_{BA},C_B\}}) \quad (3)$$

If the MAC verifies correctly, A knows that B generated the response after the request was sent. If confidentiality and authentication are needed for the original message (2), (1) can be applied [24].

2) μTESLA

μTESLA is the "micro" version of TESLA (Timed Efficient Stream Loss-tolerant Authentication) proposed by Perrig *et al.* in 2002 [30]. It emulates asymmetry through the delayed disclosure of symmetric keys and serves as the broadcast authentication service of SNEP. μTESLA relies solely on this delayed disclosure, unlike its predecessor, which authenticates the initial packet using the digital signature. It has been argued that whilst symmetric keying techniques are attractive, limitations have been exhibited in the flexibility of the symmetric key exchange protocols as a result of their energy efficiency [31].

μTESLA requires that the base station and the nodes be loosely time synchronized, and that each node knows an upper bound on the maximum error for synchronization. For an authenticated packet to be sent, the base station computes a MAC on the packet with a key that is secret at that point in time. When a node gets a packet, it can confirm that the base station did not yet disclose the corresponding MAC key, using its loosely synchronized clock, maximum synchronization error and the time at which the keys are to be disclosed. The node stores the packet in a buffer, aware that the MAC key is only known to the base station, and that no adversary could have altered the packet during transmission.

When the keys are to be disclosed, the base station broadcasts the key to all receivers. The receiver can then verify the correctness of the key and use it to authenticate the packet in the buffer [29]. Each MAC key is a member of a key chain, which has been generated by a one-way function *F*. In order to generate this chain, the sender chooses the last key, K_n, of the chain randomly, and applies *F* repeatedly to compute all other keys:

$$K_i = F(K_{i+1}) \quad (4)$$

Applying the SNEP building block, each node can easily perform time synchronization and retrieve an authenticated key from the key chain for the "commitment in a secure and authenticated manner" [24] (a detailed description of μTESLA can also be found here).

Schemes like μTESLA, based on delayed key disclosure, can suffer from denial-of-service attacks (DOS). In the subsequent interval when the message is in the buffer and the receiver waits on the disclosure time, an attacker can flood the network with arbitrary messages, claiming them to belong to the current time interval. Only in the next time interval can the nodes determine that these messages are not authentic. This type of attack can lead to buffers overflowing, and battery exhaustion, as all messages are forwarded to the nodes. The use of public key cryptography would eliminate the need for such complicated protocols, increasing the security of the system, and only requiring the public key of the base station to be embedded in the nodes [31].

3) Implementation

This is a summary of the tools used in the implementation of SPINS. The chosen block cipher was a subset of RC5, available from OpenSSL [32], chosen for its small code size and high efficiency. Further optimization allowed for an additional 40% reduction of code size.

The same function is used to perform encryption and decryption. This is the counter mode (CTR) [35] of block ciphers. As CTR is a stream cipher, the size of both the plaintext and ciphertext are the same and not multiples of the block size, leading to conservation of energy.

The addition of a nonce and request message, as previously described, allows for strong freshness for applications with this requirement. A MAC function is used as the pseudorandom number generator (PRG) with the secret random number generator key X_{rand}. The *C*-th pseudorandom output block is computed as MAC(X_{rand}, *C*).

For secure authentication, the CBC-MAC algorithm is used. The operation of this algorithm is included in Appendix A below. In this system, a MAC is computed per packet, and is suitable for this environment. The MAC checks both authentication and integrity of the packet "eliminating the need for mechanisms such as CRC" [29].

Key setup depends on a secret master key, X_{AS}, shared between the node, A, and the base station, S. All other keys in this system are bootstrapped from this secret master key, using a pseudorandom function *F*, implemented as $F_K(x) = \mathrm{MAC}(K, x)$.

4) Evaluation

Important evaluation considerations in this type of system include code size and speed of operation. SPINS can occupy from 1580 to 2674 bytes of memory, and 6.30 to 7.24 ms to complete, depending on optimization.

The memory size (RAM) required by the modules is 220 bytes. Approximately 20% of the energy cost of sending a packet secured under SPINS is consumed by the security operations.

Karlof *et al.* (2004) state that SNEP was, unfortunately, neither fully specified nor fully implemented. This meant that SPINS could not be fully adopted for use with wireless sensor networks; motivating the arrival of TinySec [33], which is integrated into TinyOS [34].

TABLE III. SPINS SECURITY CHARACTERISTICS

	Encryption	Block Cipher	Freshness	Code Requirement	Auth. Provided	Cost (time/ energy)	Key Agreement
SPINS	Yes - CTR mode	RC5	Yes	2674 Bytes Max	Yes – CBC-MAC	7.2ms/ 20 %	Master Key & Delayed Disclosure

B. TINYSEC

Karlof *et al.* (2004) designed the replacement for the unfinished SNEP, known as TinySec, a "Link Layer Security Architecture for Wireless Sensor Networks" [33]. Inherently, it provides similar services, including access control, message integrity and confidentiality. Access control and integrity are ensured through authentication, and confidentiality through encryption. Semantic security is achieved through the use of a unique initialization vector (IV) for each invocation of the encryption algorithm.

Replay protection has been intentionally omitted based on the difficulty of the task at hand with respect to the limited amount of state that each recipient in the network can keep. It is argued that replay protection belongs in the higher layers of the communication stack, and not at the link-layer [33].

TinySec allows for two specific variants. The first of these, TinySec-Auth, provides for authentication only, and the second, TinySec-AE, provides both authentication and encryption. For TinySec-Auth, the entire packet is authenticated using a MAC, but the payload data is not encrypted; whilst using authenticated encryption, TinySec encrypts the payload and then authenticates the packet with a MAC.

In contrast to SPINS, it is argued that a stream cipher is not optimal as a result of possible IV reuse, and therefore an encryption scheme with good resilience to repeated IVs should be used. Resultantly, TinySec employs a block cipher with a specific mode of operation, and the intuitive CTR mode was rejected.

1) Implementation

The MAC used for authentication in both modes of TinySec is CBC-MAC. Generally, the security of CBC-MAC is determined by the length of the MAC. In this case, a MAC of 4 bytes, considerably less than the conventional 8 or 16 bytes, is specified for use. Karlof *et al.* (2004) argue that, in the context of sensor networks, this is not detrimental [33]. Should an adversary repeatedly attempt blind forgeries, it will succeed after 2^{31} attempts. Adversaries can only assess the validity of an attempted forgery by sending it to an authorized receiver. This implies that approximately 2^{31} packets must be sent in order to forge just on malicious packet. For wireless sensor networks, this is an adequate level of security, and for an attempt similar to the one previously described, it would take up to 20 months to be successful on a 19.2 Kb/s channel.

Implicitly, an effective denial-of-service attack could be launched in this manner; as the radio channel would be locked for an extensive period of time as attempts are made, and battery power would be consumed by recipient nodes testing the validity of the received packets. It is argued that a simple heuristic, whereby the nodes signal the base station when the rate of MAC failures exceeds a predetermined threshold, would alleviate the problem [33].

A variant of CBC mode of block cipher operation is employed by TinySec. This choice is primarily due to its graceful degradation in the face of repeated IVs. The variant chosen has a fix included to the leakage problem of CBC mode when a counter is used as the IV. In order not to create ciphertext which are multiples of 8 bytes, which may increase message size and power consumption the ciphertext stealing technique (CTS) is employed [35]. This technique ensures that the ciphertext is the same length as the plaintext.

TABLE IV. TINYSEC SECURITY CHARACTERISTICS

	Encryption	Block Cipher	Freshness	Code Requirement	Auth. Provided	Cost (time / energy)	Key Agreement
TINYSEC	Optional - CBC mode	Skip-jack	No	7146 Bytes Max	Yes – CBC-MAC	0.38 ms/ 9.1%	Any

The block ciphers found to be most useful for implementation in terms of speed and size were RC5 and Skipjack [35]. Although RC5 implementation was slightly quicker, the Skipjack one was chosen for distribution as there were no patent issues, and there is no requirement for a pre-computed key schedule (saving 104 bytes of RAM per key).

There is no particular keying mechanism specified for use with TinySec. This means that any keying mechanism can be employed [36]. Suggested keying mechanisms include single network-wide keys, per-link keys between neighboring nodes and group keys. Each of these has their own specific advantages and disadvantages [33].

2) Evaluation

TinySec is implemented in approximately 3000 lines of nesC code. It requires 728 bytes of RAM and 7146 bytes of program space. Completion of the cipher operation (Skipjack) takes approximately 0.38 ms. Depending on the mode of TinySec, authentication only or encrypted authentication, an increase in energy consumption of 3.03% or 9.1%, respectively, over that of sending a normal TinyOS packet, can be seen. The increases in energy cost can be mainly attributed to the increased packet size induced through the provision of the security functions [33].

It is worth reaffirming that TinySec was distributed with official releases of TinyOS version 1.x [37]. It has proven that efficient secure communication in wireless sensor networks is a feasible reality. Additionally, it was used as link layer basis for a number of research projects, and a version was taken on in industry by Bosch for a prototype burglar alarm security system.

C. LEAP/LEAP+

Localized Encryption and Authentication Protocol (LEAP) was proposed by Zhu *et al.* (2003) as a key management protocol for sensor networks, motivated by the observation that different types of messages propagated in wireless sensor networks have different security requirements [38]. Lightweight, energy efficient operation and robustness and survivability in the face of node compromise, are the main design goals of this protocol.

A prototype implementation of LEAP (LEAP+) was implemented on the Berkeley Mica2 motes [39, 40]. This sees a number of the original design ideas made redundant, but will be discussed nonetheless. Also implemented for use with TinyOS through nesC, the protocol uses RC5 for encryption and CBC-MAC for authentication.

There are four different keying mechanisms provided by LEAP, in keeping with the need for different security requirements for different types of messages. These include Individual Keys, Group Keys, Cluster Keys and Pairwise Shared Keys. The Individual Key is a unique key that every node shares with the base station. This allows for confidential communication between the base station and individual nodes, useful for special instructions or keying material etc.

The Group Key is a globally shared key that is used by the base station for sending encrypted messages to the entire sensor network (or Group). This may be used to send queries or interests, or to propagate a mission to the nodes of the network. A Cluster Key is similar but is shared between a node and its neighbors. This is mainly employed for securing locally broadcast messages (routing information or enabling passive participation, for example) [33].

A Pairwise Shared Key is a key which every node shares with each of its immediate neighbors. These keys are used under this scheme for secure communications that need privacy or source authentication. It could also use this key to distribute a Cluster Key, for example. The use of these keys precludes passive participation. The actual establishment of keys is somewhat outside the scope of this research, but a detailed explanation can be found in [38].

An attractive property of this security architecture is the manner in which it handles inter-node authentication. μTESLA is used for broadcast authentication, but between nodes, one-way key chain based authentication is used. Every node creates a one-way key chain of a certain length, and transmits the first key of the chain to each neighbor (encrypted with the Pairwise Shared Key). A key from the one-way chain is known as the AUTH key, and whenever a node sends a message, it attaches the next AUTH key in the chain. The keys are disclosed in reverse order to their generation, and a receiver can verify the authenticity of the message based on the received initial key (commitment), or a recently disclosed AUTH key.

In this manner, packets are authenticated on a hop-by-hop basis by immediate neighbors. Also, nodes will usually receive the packet before it could receive a copy of the packet forwarded by another node, resulting from the triangular inequality of the distances between the involved nodes. This means that an adversary could reuse a key from the chain to impersonate a particular node without being recognized, and copies of messages can be discarded. There remain a number of possible attacks to this scheme, but are addressed by Zhu *et al.* [38].

1) Implementation

TABLE V. LEAP/LEAP+ SECURITY CHARACTERISTICS

	Encryption	Block Cipher	Freshness	Code Requirement	Auth. Provided	Cost (time / energy)	Key Agreement
LEAP	Yes - RC5	RC5	No	17.8 Kb	Yes – CBC-MAC	Variable (No. of neighbors)	Pre-deployed (Master) Variable

A description of the implementation of LEAP+ is available in [39]. It is implemented on the Berkeley Mica2 motes through TinyOS. The implementation retains the functionality described for Pairwise key, Cluster key and one-way key chain generation, but does not include the scheme for re-keying the global key or broadcast source authentication based on μTESLA.

Encryption and authentication (CBC-MAC) are achieved by using the RC5 block cipher, specified by Rivest in 1995 [41]. As RC5 provides all of the security primitives for LEAP+, the functions for deriving master and pairwise keys, and the one-way functions for producing one-way key chains are replaced by MAC with RC5. Under this scheme, a new node needs to send one message to establish a pairwise key and another for the cluster key. A standard 8 byte key length is specified for use in the protocol [39].

2) Evaluation

This security architecture now has many similar attributes to its predecessors. It is implemented, written in nesC code for TinyOS, in 17.8 Kb of memory (ROM), and the RAM usage is determined by the number of neighbors present for each node. For one node it uses 600 bytes, for thirty nodes it uses 1.59 Kb. This is reasonable usage based on the 128 Kb of program memory and 4 Kb of RAM provided by the Mica2 mote. In this prototype implementation, a node can complete determination of pairwise keys with three neighbors in approximately 8.5 seconds.

The authors consider a number of possible attacks to the security of the protocol (including spoofing, replay, HELLO flood, wormhole and sinkhole attacks), and argue that LEAP is robust against them, whilst detection of any of these attacks is relatively simple [39]. They also intend to fully implement the protocol and contribute this to the TinyOS community [42].

It is claimed that it's key sharing approach supports in-network processing, while restricting the impact of a compromised node to the immediate network, and can prevent or increase the difficulty of launching many attacks to the security of wireless sensor networks [39].

D. Security Manager

Heo and Hong (2006) proposed a new method of authenticated key agreement [16]. It is based on a Public Key Infrastructure (PKI) and Elliptic Curve Cryptography (ECC). The Security Manager (SM) gives static domain parameters such as the base point and elliptic curve coefficients to prospective network nodes. Devices use these initial parameters to establish

permanent public keys and ephemeral public keys, which are in turn used for securing the network data. After calculating a public key, a node sends this to the SM, which could have a public key list for all nodes in the network.

ECC is an approach to public-key cryptography which is based on the algebraic structure of elliptic curves, over finite fields [43]. Elliptic Curve algorithms provide reasonable computational loads and smaller key sizes for equivalent security to other techniques. Smaller keys sizes reduce the size of message buffers and implementation cost of protocols. Authenticated key agreement is achieved via the SM, based on the EC-MQV algorithm [16]. This algorithm is more advanced than Diffie-Hellman [44], eliminating the man-in-the-middle attack. Diffie-Hellman is included in the EC-MQV algorithm as a subset [45].

This approach is based on the IEEE 802.15.4 standard, and the responsibilities of the SM are carried out by the LR-WPAN coordinator, as specified in [12]. Using a pre-deployed recognition function, the coordinator acts as the Security Manager and distributes the initial trust parameters to recognized nodes. The overhead costs involved in this process will depend largely on the number of bits chosen for the elliptic curve. An elliptic curve cryptosystem can provide the same security as RSA using a 160 bit key versus 1024 bit key respectively. RSA is a traditional method for public key cryptography [46]; one which many find too expensive for wireless sensor network applications. The key length (a parameter) can be varied, allowing for an increased level of security for a larger key length, implying a variable level of overhead.

One of the major requirements of a system like this is resiliency to SM failure. Should the SM be lost or stolen (i.e. through battery failure or physical theft, respectively) the network should be able to recover and continue seamlessly. This requirement is addressed in the IEEE 802.15.4 standard, for which this architecture is proposed.

1) Implementation

A prototype implementation of this proposed mechanism is unavailable to date.

2) Evaluation

The scalable nature of this solution is attractive for use with wireless sensor networking, as it can be increased or reduced depending on the nature of the disseminated message. A concern would be the number of keys the SM would be required to store as the network grows, and every node may not come into range of the SM. This can be addressed by allowing more than one node to have SM responsibilities.

TABLE VI. SM SECURITY CHARACTERISTICS

	Encryption	Block Cipher	Freshness	Code Requirement	Auth. Provided	Cost (time / energy)	Key Agreement
SM	Yes – ECC	N/A	No	N/A	EC-MQV	Variable (Nodes, parameters etc.)	EC-MQV Initial trust

The trade-off between power conservation will still apply, but may be significantly less than other public key architectures.

E. ZigBee

As previously stated, the ZigBee specification outlines the design of the NWK layer that operates just above the PHY and MAC layers specified by the IEEE802.15.4 standard. Additionally, it contains descriptions, protocols and algorithms relating to the application support layer (APS), ZigBee device objects (ZDO) and profile (ZDP), the application framework and ZigBee security services [13].

The concept of a "Trust Center" is introduced in the specification. Generally, the ZigBee coordinator performs this duty. The coordinator allows other devices to join the network and distributes the appropriate keying information. There are three roles played by the "Trust Center"; 1: trust manager, whereby authentication of devices requesting to join the network is carried out, 2: network manager, maintaining and distributing network keys, and 3: configuration manager, enabling end-to-end security between devices [47]. There are two modes of operation; Residential Mode and Commercial Mode. Running the former, low security residential applications are accounted for. The latter is designed for high-security commercial applications.

In Residential Mode, the Trust Center will allow devices to join the network, but does not establish keys with the network devices. It therefore cannot periodically update keys and allows for the memory cost to be minimal, as it cannot scale with the size of the network. In Commercial Mode, it establishes and maintains keys and freshness counters with every device in the network, allowing centralized control and update of keys. This results in a memory cost that could scale with the size of the network [47]. This could be managed through means clustering, for example.

There are three types of keys specified for use in ZigBee security services; the Master Key, the Link Key and the Network Key. Master keys are installed first, either in the factory or out of band. They are sent from the Trust Center and are the basis for long-term security between two devices. The Link Key is a basis of security between two devices and the Network Keys are the basis of security across the entire network. Link and Network Keys, which are installed either in the factory or out of band, employ symmetrical key-key exchange (SKKE) handshake between devices. The key is transported from the Trust Center for both types of keys. This operation occurs only in Commercial Mode, as Residential Mode does not allow for authentication.

The ZigBee specification states that CCM* mode of operation is used for security services. CCM* is a generic combined encryption and authentication block cipher mode. This mode of operation is detailed in the ZigBee specification [13], and is a combination of CTR mode and CBC-MAC mode. It differs only slightly from CCM mode [48], by offering encryption-only and integrity-only capabilities.

There are underlying levels of security supported by this architecture, and can be varied depending on the amount and type of security the data is required to maintain. The first of these modes offers no security, i.e. packets sent are void of any security. Encryption only allows for packets to be sent in an encrypted manner only. Packets can be sent in an authenticated manner only, and depending on the level of security desired, a MAC of 4, 8 or 16 bytes can be used. Finally, encrypted authenticated packets can be sent, again varying the level of protection by employing a MAC of 4, 8 or 16 bytes. In total, there are eight underlying levels. AES 128 is the specified block cipher, as it appears in NIST FIPS Pub 197 [18], and retains the block size of 128 bits throughout.

Only parties that possess the symmetric key should be able to compute the MAC. The MAC protects packet headers in addition to the data payload. The sender appends the plaintext data with a MAC. The recipient can verify the MAC by computing the MAC and comparing it with the value received in the packet. Initially, CCM* mode applies integrity protection over the header and data payload using CBC-MAC, and then encrypts the data payload and MAC using AES-CTR mode. In this way, AES-CCM includes the fields from both the authentication and encryption operations (a MAC and the frame and key counters), which serve the same functions as described above [49].

1) Implementation

To date, open source ZigBee implementations do not yet provide security support [50]. Commercially available implementations of the ZigBee stack claim to have fully functioning security, such as the EmberZNet Protocol Stack [51]. Texas Instruments provide the Z-Stack, free of charge which is compatible with CC2430 or the CC2420 transceiver, with the MSP430 microcontroller [52]. This is available in the form of the Tmote Sky, from the Moteiv Corporation [53]. The performance of these motes using ZigBee, with and without security invoked, is currently under examination.

In Commercial Mode, security can be easily invoked as a 3 bit, the security level subfield, is set. The security level identifier '000' is set for no security; whereas '111' implies that the security requirement is encrypted authentication with a 16 byte MAC (the highest available) [13].

2) Evaluation

The scalability of the security suites in this architecture is ideal for developing the technology, as implementations that do/do not require high levels of protection can be developed using the same platform. It is also useful to applications which may need to send some messages with higher security than others.

Implicitly, there will be increases in both the message latency and power consumption of the motes the higher the security requirement. Considering that security is integrated into the ZigBee protocol, it not worth considering extra code space required for implementation etc. Power consumption and latency are currently under investigation, but these trade-offs in relation to security provided will not prove detrimental to the application and usefulness of ZigBee in the wireless sensor networking domain.

TABLE VII. ZIGBEE SECURITY CHARACTERISTICS

	Encryption	Block Cipher	Freshness (CTR)	Code Requirement	Auth. Provided	Cost (time / energy)	Key Agreement
ZigBee	Yes - AES	AES-128	Yes – CCM*	N/A	Yes-CBC-MAC	Under Review	SKA Trust Center

V. RELATED WORK

Early work pertaining to ECC in wireless sensor networks was EccM, described by Malan *et al*. (2004) [54]. Based on the Mica2 mote, a public key infrastructure was implemented and evaluated, viable for TinySec keys, through elliptic curve cryptography. A need for some public key infrastructure was illustrated and proven feasible on the motes, under TinyOS.

Cryptanalysis has begun within the realm of wireless sensor networks. This is illustrated by Finnigin *et al*. (2007) [55]. They specifically targeted the previously described EccM, and found that they could easily break this algorithm within 25 minutes, or as little as 2 minutes, depending on the mote's address space, through a weakness found in the pseudorandom number generator (PRNG). A solution to the problem is also proposed, allowing security against similar attacks for up to 6.6 years on average, using an improved PRNG in the beta release of TinyOS 2.0.

TinyECC is another variation of elliptic curve cryptography for TinyOS [56]. It supports a number of motes including the MICAz, and supports all elliptic curve operations over F_P. This is in contrast to EccM, above, which operates over F_{2P}. Evaluation and documentation of the system can be found on the authors' website [56].

Wander *et al*. (2005) present an analysis of energy consumption for public key cryptography in WSNs [57]. Their results also indicate that public key infrastructures are viable for use in wireless sensor networking, and that the benefits of transmitting the smaller ECC keys and certificates will be significant in improving energy conservation.

Returning to symmetric schemes, the various block ciphers and modes of operation are surveyed and critiqued by *Law et al.* (2006) in [58]. Based on evaluation results, it is concluded that Skipjack, MISTY1 and Rijndael are the most suitable ciphers for use in WSNs, while Output Feedback Mode (OFB) and CBC modes are the recommended modes of operation (for pairwise links and group communications respectively).

Vitaletti and Palombizio (2007) critique the appropriateness of Rijndael (AES) for wireless sensor networks [59], in relation to speed, claiming that memory usage and energy efficiency should be the main concerns.

TABLE VIII. SECURITY ARCHITECTURE COMPARISON TABLE

	Encryption	Block Cipher	Freshness (CTR)	Code Requirement	Auth. Provided	Cost (time / energy)	Key Agreement	Release Year
SPINS	Yes - CTR mode	RC5	Yes	2674 Bytes Max	Yes – CBC-MAC	7.2ms/ 20 %	Master Key & Delayed Disclosure	2002
TINYSEC	Optional - CBC mode	Skip-jack	No	7146 Bytes Max	Yes – CBC-MAC	0.38 ms/ 9.1%	Any	2004
LEAP/LEAP+	Yes - RC5	RC5	No	17.8 Kb	Yes – CBC-MAC	Variable (No. of neighbors)	Pre-deployed (Master) Variable	2003 (Implemented 2006)
SM	Yes - ECC	N/A	No	N/A	EC-MQV	Variable (Nodes, parameters etc.)	EC-MQV Initial trust	2006
ZigBee	Optional - AES	AES-128	Yes – CCM*	N/A	Yes – CBC-MAC	Under Investigation - Expected increase in latency and power consumption	SKA Trust Center	2005

VI. CONCLUSION

Table VIII above illustrates the various characteristics of the previously discussed security architectures under common headings. As is evident, Symmetric key cryptography based architectures have been the main source of security in Wireless Sensor Networking to date. There is much research available claiming that Public Key based solutions will provide better solutions, based on smaller key sizes and less storage requirements (under ECC), for more secure communications, also even providing superior energy efficiency.

From an authentication perspective, the CBC-MAC algorithm is the most popular method of providing authentication for symmetric key based algorithms. Table VIII above illustrates that it is chosen in all implementations of security architectures for WSNs to date using SKA.

Included in the table is the year of release for each of these schemes. This illustrates the progression in the field, and suggests that it will continue for years to come. The trend seems to be shifting in the direction of ECC and PKIs, and much more research into this can be expected (Refer to Section V).

From a design perspective, scalability of security architectures is a desirable feature. Not all applications of WSNs will require the same security, and even in applications that do, different types of messages will require different degrees of security. This is a feature of the IEEE 802.15.4/ZigBee security architecture, and is a salient feature of ECC based architectures.

What is certain is that security is going to play a pivotal role in the eventual ubiquity of wireless sensor networks, and therefore will continue to be a major research area for the foreseeable future.

VII. FUTURE WORK

Continued research into the suitability and efficiency of security architectures for WSNs will be carried out through both implementation and simulation. It is envisaged that in order for WSNs to obtain widespread use in the public domain, standardization will have to occur. Consider the many plug-and-play ZigBee based solutions. An evaluation of the security provided by this specification, in terms of energy efficiency/network lifetime and code/memory efficiency, will be carried out, with a view to optimization and creation of further techniques to enhance the scalability of its security architecture; not specific to ZigBee. The focus is on making the scheme suitable for all applications of WSNs, whether security dependant or not, in a manner accessible to all network designers.

APPENDIX A CBC-MAC OPERATION

Due to the popularity of the use of CBC-MAC to provide authentication in the security architectures of WSNs, its mode of operation is included in this appendix:

Let $E_K(X)$ denote the enciphered n-bit block X using key K and block cipher E. For completeness, assume that E = AES, implying that n = 128. Let $a \oplus b$ denote the bitwise exclusive-or of a and b. Let $(a \,||\, b)$ denote the concatenation of strings a and b and let $|a|$ denote the length of, in bits, of a. Let 0^i denote i zero-bits.

To authenticate with the basic CBC-MAC, one begins with a message M whose length is a positive multiple of n, and a key K for E. Let:

$$M_1 \,||\, M_2 \,||\, \cdots \,||\, M_m = M \text{ with } |M_i| = n \text{ for } \quad 1 \le i \le m.$$

Then the CBC-MAC of M is defined as C_m, where

$$C_i = E_K(M_i \oplus C_{i-1}) \text{ for } 1 \le i \le m, \text{ and } C_0 = 0^n.$$

ACKNOWLEDGMENT

This work was supported by Science Foundation Ireland – Grant Number 05/RFP/CMS0071.

REFERENCES

[1] Boyle, D., Newe, T., 2007. 'Security Protocols for use with Ad-Hoc Wireless Sensor Networks: A Survey of Security Architectures'. *IEEE Third International Conference on Wireless and Mobile Communications - ICWMC 2007*. 4th – 9th March 2007, Guadeloupe, French Caribbean, ISBN 0-7695-2796-5.

[2] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., Cayirci, E. (2002) 'A Survey on Sensor Networks', *IEEE Communications Magazine*, 40(8), 102-114.

[3] Bharathidasan, A., Anand, V., Ponduru, S. (2001), Sensor Networks: An Overview, Department of Computer Science, University of California, Davis 2001. Technical Report

[4] Crossbow Technologies Inc. (2007) *MICAz* [online], available: http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/6020-0060-01_A_MICAz.pdf [accessed 7 Dec 2007].

[5] Chee-Yee Chong, and Kumar, S. P. (2003), "Sensor Networks: Evolution, Opportunities, and Challenges", Proceedings of the IEEE, Vol. 91, No. 8, August 2003: IEEE, 1247-1256.

[6] Jeong, J., Jiang, X. F. and Culler, D. E. (2007), Design and Analysis of Micro-Solar Power Systems for Wireless Sensor Networks, Electrical Engineering and Computer Sciences, University of California at Berkeley, 2007. Technical Report

[7] Perrig, A., Stankovic, J., Wagner, D. (2004), "Security in Wireless Sensor Networks", *Communications of the ACM*, 47(6), 53-57.

[8] Garcia-Hernandez, C. F., Ibarguengoytia-Gonzalez, P. H. and Perez-Diaz, J. A. (2007) 'Wireless Sensor Networks and Applications: A Survey', *IJCSNS International Journal of Computer Science and Network Security*, 7(3), 264-273.

[9] Defense Advanced Research Projects Agency (13 Oct 2006) Defense Advanced Research Projects Agency Home [online], available: http://www.darpa.mil/index.html [accessed 26 July 2007].

[10] Jovanov, E., Milenkovic, A., Otto, C., de Groen, PC. (2005) 'A Wireless Body Area Network of Intelligent Motion Sensors for Computer Assisted Physical Rehabilitation', Journal of Neuroengineering and Rehabilitation, 2(6).

[11] Bluetooth SIG (2006) *How Bluetooth Technology Works* [online], available: http://www.bluetooth.com/Bluetooth/Learn/Works/ [accessed 7 Dec 2007]

[12] IEEE 802.15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs) (2003), 3 Park Avenue, New York, USA: IEEE.

[13] ZigBee Specification v1.0: ZigBee Specification (2005), San Ramon, CA, USA: ZigBee Alliance.

[14] Boyle, D., Newe, T., 2007. "The Magnitude of Communications Protocols in Wireless Sensor Networks". *WLANMESH – Towards an 802.11 standard for Mesh Networking*. 17th – 20th April 2007, Paris, France.

[15] Xiao, Y., Chen, H-H., Sun, B., Wang, R., and Sethi, S. (2006) 'MAC Security and Security Overhead Analysis in the IEEE 802.15.4 Wireless Sensor Networks', *EURASIP Journal on Wireless Communications and Networking*, 2006, 1-12.

[16] Heo, J., Hong, C.S. (2006) "Efficient and Authenticated Key Agreement Mechanism in Low-Rate WPAN Environment", *International Symposium on Wireless Pervasive Computing 2006*, Phuket, Thailand 16 – 18 January 2006, IEEE 2006, 1-5.

[17] Sastry, N. and Wagner, D. (2004) 'Security Considerations for IEEE 802.15.4 Networks', *Proceedings of the 2004 ACM Workshop on Wireless Security*, Philadelphia, PA, USA, October 1, 2004, New York, USA: ACM Press, 32-42.

[18] Federal Information Processing Standards Publication 197: Advanced Encryption Standard (AES) (2001), USA: National Institute of Standards and Technology (NIST).

[19] Hill, J. L., (2003) System Architecture for Wireless Sensor Networks, PhD, University of California, Berkeley.

[20] Levis, P., Madden, S., Gay, D., Polastre, J., Szewczyk, R., Woo, A., Brewer, E., Culler, D. (2004) 'The Emergence of Networking Abstractions and Techniques in TinyOS', *Proceedings of the First Symposium on Networked Systems Design and Implementation*, 29th-31st March, 2004, San Francisco, CA, USA.

[21] Gay, D., Levis, P., von Behren, R., Welsh, M., Brewer, E. and Culler, D. (2003) 'The *nesC* Language: A Holistic Approach to Network Embedded Systems', *PLDI Conference on Programming Language Design and Implementation*, San Diego, California, USA, 09-11 June 2003, New York, USA: ACM Press, 1-11.

[22] UC Berkeley (2004) *Packet Protocols: TEP 116* [online], available: http://tinyos.cvs.sourceforge.net/*checkout*/tinyos/tinyos-2.x/doc/html/tep116.html [accessed 7 Dec 2007].

[23] Newsome, J., Shi, E., Song, D., Perrig, A (2004) 'The Sybil Attack in Sensor Networks: Analysis and Defenses'. *Proceedings of the Third international Symposium on Information Processing in Sensor Networks*, 26-27 April 2004, Berkeley, California, USA.

[24] Deng, J., Han, R., Mishra, S. (2004) 'Intrusion Tolerance and Anti-Traffic Analysis Strategies for Wireless Sensor Networks', *The International Conference on Dependable Systems and Networks, 1 July*, 2004, Florence, Italy.

[25] Bryan Parno, Adrian Perrig, Virgil Gligor, "Distributed Detection of Node Replication Attacks in Sensor Networks," *sp*, pp. 49-63, 2005 IEEE Symposium on Security and Privacy (S&P'05), 2005

[26] Karlof, C., Wagner, D (2003) "Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures", *Ad-Hoc Networks* 1(3), 293-315.

[27] Sarma, H., Kar, A. (2006) 'Security Threats in Wireless Sensor Networks', *2006 International Carnahan Conference on Security Technology*, 16th-20th October 2006, Kentucky, USA.

[28] Liu, D., Ning, P., Zhu, S., Jajodia, S. (2005) 'Practical Broadcast Authentication in Sensor Networks', the Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'05), San Diego, California, USA, 17-21 July 2005: IEEE Computer Society Press, 118-132.

[29] Perrig, A., Szewczyk, R., Tygar, J.D., Wen, V. and Culler, D (2002) 'SPINS: Security Protocols for Sensor Networks', Wireless Networks, 8(5), 521-534.

[30] Perrig, A., Canetti, R., Tygar, J.D. and Song, D. (2002) 'The TESLA Broadcast Authentication Protocol', CryptoBytes, 5(2), 2-13.

[31] Kaps, J. -P. (2006) Cryptography for Ultra-Low Power Devices, unpublished thesis (PhD), Worcester Polytechnic Institute.
[32] SpaceNet AG (2007) *OpenSSL* [online], available: http://www.openssl.org/ [accessed 7 Dec 2007].
[33] Karlof, C., Sastry, N., Wagner, D. (2004) 'TinySec: A Link Layer Security Architecture for Wireless Sensor Networks', Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, Baltimore, MD, USA, 03 – 05 November 2004, New York, NY, USA: ACM Press, 162 – 175.
[34] UC Berkeley (2004) *TinySec: User Manual* [online], available: http://www.tinyos.net/tinyos-1.x/doc/tinysec.pdf [accessed 7 Dec 2007]
[35] Schneier, B. (1996) '*Applied Cryptography: Protocols, Algorithms, and Source Code in C*', 2nd ed., USA: Wiley.
[36] Camtepe, S. A., Yener, B. (2005), Key Distribution Mechanisms for Wireless Sensor Networks: A Survey, Department of Computer Science, Rensselaer Polytechnic Institute, New York, 2005. Technical Report.
[37] UC Berkeley (2003) *TinyOS Documentation* [online], available: http://www.tinyos.net/tinyos-1.x/doc/ [accessed 7 Dec 2007]
[38] Zhu, S., Setia, S., Jajodia, S. (2003) 'LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks', CCS '03, Washington D.C., USA, 27 – 31 October 2003, New York, USA: ACM Press, 62-72.
[39] Zhu, S., Setia, S., Jajodia, S. (2006) 'LEAP+: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks', *ACM Transactions on Sensor Networks TOSN*, 2(4), 500-528.
[40] Crossbow Technology Inc (2007) *MICA2* [online], available: http://www.xbow.com/Products/productdetails.aspx?sid=174 [accessed 7 Dec 2007]
[41] Rivest, R. (1994) 'The RC5 Encryption Algorithm', *Fast Software Encryption LNCS 1008*, Preneel, B., Ed., Springer-Verlag, 1995, 86-96.
[42] UC Berkeley (2007) *TinyOS Community Forum: an Open Source OS for the Networked Sensor Regime* [online], available: www.tinyos.net [accessed 7 Dec 2007].
[43] Koblitz, N., (1987) 'Elliptic Curve Cryptosystems', *Mathematics of Computation 48*, 1987, 203-209.
[44] Diffie, W. and Hellman, M. E., (1976) 'New Directions in Cryptography', *IEEE Transactions on Information Theory*, vol. IT-22, Nov. 1976, 644-654.
[45] Menezes, A., Qu, M., Vanstone, S. (1995) 'Some new key agreement protocols providing implicit authentication', *Proceedings of Workshops on Selected Areas in Cryptography*, 1995.
[46] RSA Laboratories (2007) *PKCS #1: RSA Cryptography Standard* [online], available: http://www.rsa.com/rsalabs/node.asp?id=2125 [accessed 7 Dec 2007].
[47] ZigBee Alliance (2006) *ZigBee Security Specification Overview* [online], available: http://www.zigbee.org/en/events/documents/december2005_open_house_presentations/zigbee_security_layer_technical_overview.pdf [Accessed 7 Dec 2007].
[48] Jonsson, J., (2002) 'On the Security of CTR + CBC-MAC', *Proceedings of Selected Areas in Cryptography – SAC 2002*, Nyberg, K., Heys, H., eds., Lecture Notes in Computer Science, Vol. 2595, 76-93.
[49] D. Whiting R. Housley and N. Ferguson, "Counter with CBC-MAC (CCM)", RFC 3610, Internet Eng. Task Force, Sept. 2003.
[50] Open-zb.net (2006) *Open-Source Toolset for IEEE 802.15.4 and ZigBee* [online], available: http://www.open-zb.net/ [accessed 7 Dec 2007].
[51] Ember Corporation (2007) *ZigBee Software* [online], available: http [accessed 15 Aug 2007].
[52] Texas Instruments Incorporated (2007) *Z-Stack ZigBee Protocol Stack* [online], available: http://focus.ti.com/docs/toolsw/folders/print/z-stack.html [accessed 15 Aug 2007].
[53] Moteiv Corporation (2007) *Tmote Sky* [online], available: http://www.moteiv.com/products/tmotesky.php [accessed 15 Aug 2007].
[54] Malan, D. J., Welsh, M., Smith, M. D. (2004) 'A Public-Key Infrastructure for Key Distribution in TinyOS Based on Elliptic Curve Cryptography', in Znati, T. chair *First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, IEEE SECON 2004*, Santa Clara, California, USA, 4-7 October 2004, p 71-80.
[55] Finnigin, K. M., Mullins, B. E., Raines, R. A. and Potoczny, H. B. (2007) 'Cryptanalysis of an elliptic curve cryptosystem for wireless sensor networks', *Int. J. Security and Networks*, Vol. 2, Nos. 3/4, pp.260-271.
[56] Ning, P. (2007) *TinyECC: Elliptic Curve Cryptography for Sensor Networks* [online], available: http://discovery.csc.ncsu.edu/software/TinyECC/ [accessed 7 Dec 2007].
[57] Wander, A., Gura, N., Eberle, H., Gupta, V., Shantz, S. C. (2005) 'Energy Analysis of Public-Key Cryptography for Wireless Sensor Networks', *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications PerCom2005*, Kauai Island, Hawaii, March 8-12, 2005, IEEE pp 324-328.
[58] Law, Y. W., Doumen, J. and Hartel, P. (2006) 'Survey and Benchmark of Block Ciphers for Wireless Sensor Networks', *ACM Transactions on Sensor Networks TOSN*, 2(1), 65-93.
[59] Vitaletti, A., Palombizio, G. (2007) 'Rijndael for Sensor Networks: Is Speed the Main Issue?' *Electronic Notes in Theoretical Computer Science ENTCS*, 171(1), 71-81.

David Boyle was born in Limerick, Ireland in 1983. He received his Bachelor of Engineering degree in Computer Engineering from the University of Limerick, Limerick, Ireland in 2005. His major field of study is authentication and cryptographic protocols for wireless sensor networks.

He is currently a PhD student at the University of Limerick, Limerick, Ireland, working under the supervision of Dr. Thomas Newe. His research interests include authentication and cryptographic protocols, wireless sensor networks, communications systems for sensor networks and applications of these networks.

Mr. Boyle is a student member of the IEEE.

Thomas Newe (BEng., PhD, MIEEE) was born in Westmeath, Ireland in 1967. He received his Honours Degree in Computer Engineering in 1991. Following this he worked as a software engineer with MAC, Ireland. He was awarded his PhD in 2003 from the University of Limerick for work on Formal Verification Logics for use in security protocol design. He is currently a Lecturer in the Department of Electronic and Computer Engineering in the University of Limerick and a member of the Optical Fibre Sensors Research Centre.

Currently his major areas of research include: Wireless Sensor Networks, Operating Systems, Cryptography/Encryption Algorithms, Data Security, Network Security and Formal Verification Methods.