Application of Synchronous Dynamic Encryption System (SDES) in Wireless Sensor Networks*

Hamdy S. Soliman and Mohammed Omari
(Corresponding author: Hamdy S. Soliman)

Department of Computer Science, New Mexico Institute of Mining and Technology
Socorro, NM, USA. (Email: {hss, omari}@nmt.edu)

(Received Aug. 19, 2005; revised and accepted Oct. 1, 2005)

Abstract

Inherent to the wireless sensor networks are the two major problems of the broadcasting vulnerability, the limited computational capability and power budget. Even though security is a must in most applications, current sophisticated security protocols are not amenable to the primitiveness of the sensors. In this paper, we introduce a novel security protocol for wireless network of sensors that is very secure, yet simple and efficient. At the core of our security protocol is a simple and fast stream cipher cryptosystem that utilizes permutation vectors as encryption keys, forcing an intruder to a brute-force time complexity of $\Omega(2^n)$. In addition, our mechanism alleviates the effect of sensor capture, via its synchronized re-keying feature. In addition to the encryption efficiency, our system utilizes the group deployment of newly joining sensors for sensors power budgeting considerations. Experimental results show very promising future of our system in the wireless networks domain, excelling over other peers of modern cryptosystems (AES, DES, TripleDES), especially in the power budget arena.

Keywords: Deployment knowledge, encryption permutation vectors, power balancing, sensors security primitives, stream ciphers

1 Introduction

In addition to its inherent broadcasting nature, wireless sensor networks are characterized by their ad-hoc nature, with primitive and unguarded units, which make it very difficult to secure. Sensors routing remained for sometime as the central issue since the network topology is very dynamic. Yet, lately the importance of security in such networks has been realized [26]. On the contrary of fixed infrastructures, sensors communications cannot rely on the online availability of a centralized security protocols, such as the 802.11 WLAN standard. Therefore, the IEEE 802.15.4 Low-rate Wireless Personal Area Network Standard [14] came to specify the physical layer and medium access control layer of a low data rate, ultra low power, and low cost sensor network. Target applications include natural disaster control, health care, battlefield service, oil site operation, rescue missions, etc. In these security-sensitive deployments, secure and fast transmission of sensitive digital information over the sensor network is essential.

Sensors networks may be highly versatile, involving short-lived communications between devices that may never have communicated before, with the overhead of initial required authentication. Well-designed security services can contribute to the reliability and robustness of the sensor network's communication infrastructure and to the integrity of its transmitted data. Recent security research for ad hoc networks seemed to focus on distributing the role of the central certifying authority over some or all devices in the network [17, 29], the main approach being based on threshold cryptography [24] and allowing specific coalitions of devices to act together as a source of trust such as a certificate authority (e.g., to generate public-key certificates). Public-key operations are quite expensive though, which remains a problem for portable devices with limited computation resources and power supplies [13]. In recent years, symmetric-key based key management protocols have gained popularity due to the small encryption computation overhead [27]. However, the key establishment for pure symmetric-key in large sensor networks is a complicated process; either via online key distribution center (KDC) or pre-loading a large number of symmetric keys prior to the deployment of sensors. In addition to incurring communication overhead by the former and consumption of the scarce sensor memory by the later, both methods reduce the scalability of self-organizing sensor networks.

Another important class of encryption techniques consists of the stream cipher algorithms that encrypt individual characters (usually binary digits) of a plaintext

*The security algorithm and its applications are filed to the US patent office on December 2003.

message one at a time, using an encryption transformation that varies with time [28]. In contrast, block ciphers tend to simultaneously encrypt groups of characters of a plaintext message using a fixed encryption transformation. Stream ciphers are generally faster than block ciphers (public or symmetric key cryptosystems) [19], and have less complex hardware circuitry, thus they are used to enhance the speed of encryption while maintaining a high level of security. Hence, our security design approach follows the steps of stream ciphering. At its core, there are few XORs and additions operations, so it can achieve better performance with smaller code size and lesser complexity than standard encryption algorithms. Specifically, its streaming key management process involves, among other parameters, the use of the last generated key to generate the next key in the key stream, avoiding key duplication and protecting against replay and cryptanalysis attacks.

In this paper, we will investigate the deployment of our proposed security model, Synchronous Dynamic Encryption System (SDES) [25], for sensor networks. Section 2 describes the constraints and limitations that generally characterize sensors networks. The dilemma of the secret key establishment in sensors networks is described in Section 3. The first approach of solving the secret key distribution problem is depicted in Section 4. Section 5 explains the necessity of utilizing symmetric key cryptography for better power consumption efficiency. The advantages of stream ciphers and their easy deployment in sensor networks are stated in Section 6. Our SDES is presented in Section 7, with full details on permutation vector generation, encryption/decryption functions and secret key management. A comparison between our security model and other peer models is presented in Section 8. Section 9 is our conclusion.

2 Sensor Networks Constraints

The major two inherent constraints of typical network of sensors are namely i) its ad-hoc nature, ii) the low budget of the sensing devices that reflects on its storage, computational, transmission reach, and power capabilities. Next, We will discuss the three limitations imposed by such inherent properties and their influence on the installed security primitives.

Limited memory space. In large-scale sensor networks, thousands of nodes, it is nearly impossible for every sensor to store a shared key with every other communicating sensor in the network, as it requires large memory that is not available in the sensor. Therefore, current key establishment schemes are not quite feasible. In addition, installed security primitives have to be simple and optimized in order to fit in the sensor tiny memory. The memory size limitation also affects the storage of previous knowledge of all possible nodes that will join the network.

Limited power. Any deployed security mechanism has to minimize computation and transmission overheads in order to utilize the sensor's battery efficiently. Complicated encryption techniques (e.g. public key algorithms) and high-challenging authentication mechanisms are usually avoided in any sensor security scheme, due to power limitations.

Limited budget. Sensors are an order of magnitude lesser in price than personal computers (few dollars). Unattended sensors are highly vulnerable to be captured by intruders. A captured sensor might reveal essential info about the deployed security mechanism. Hence, we have to consider such attacks while designing sensor security mechanism in order to limit their effect in the overall integrity and performance of the network.

3 Secret Key Distribution

Essential to sensor networks that use symmetric key approach is the integrity and efficiency of distributing and sharing the secret key among the network nodes. The deployment of a central authority as a third party in the secret key establishment is impractical for large scale sensor networks because of the unknown network topology prior to deployment, communication range limitations, and network dynamics [9]. Most of the security schemes rely on key pre-distribution [9]; keys would have to be pre-installed in sensor nodes to accommodate secure connectivity between nodes. Consequently, any key establishment scheme should be somehow in the middle between two extremes: *single mission key* scheme, or the *mesh* scheme [9]. In the former, all sensor nodes share the same key, saving memory space and computational/communication electric power. However, the capture of any sensor will jeopardize the entire network security. In the mesh scheme, every sensor shares a unique secret key with all sensors in the network. A great advantage of the mesh scheme with regard to sensor capture is that all communications that do not involve the captured sensor are not compromised. However, this solution requires storing a large number of keys at each sensor. Moreover, it forces key redistribution upon any topology change, hence degrading network scalability.

Due to the high possibility and severity of node capturing, a sensor network has to be able to detect any such attack and notify all neighboring node to revoke their shared keys with the captured sensor. Moreover, at the network level, all keys that are identical/related to the revoked keys or generated through the captured node should be revoked too. Therefore, key revocation schemes should be optimal in terms of space, communication, and computation overheads.

4 Public Key Based Security

Public key cryptosystems are suitable for networks of decentralized architecture. In sensor networks, due to the huge overhead of involving a central authority, adjacent sensors mutually authenticate without the need to deploy

a third party. Thus, the use of Public-Key Cryptography (PKC) is most amenable for use in decentralized sensor networks and would eliminate the centralized authority problem [8]. Because of its asymmetry property, sensors do not need to carry the pre-distributed keys. It should be clear that an intrusion on one sensor will not affect the security of others.

One of the security threats that could beat any PKC is the man in the middle attack. The most famous solution against this type of attacks is the key certificate. When a node, say A, attempts to authenticate itself to another node B, A sends its public key with a certificate that authenticates the key itself. This certificate is encrypted by the private key of some authority. Then, B verifies the key's validity by decrypting the certificate using the authority's public key.

The most common criticism on using PKC in sensor networks is its computational complexity and communication overhead [8]. Recently, a number of studies have been conducted to evaluate the practicality of using PKC in sensor networks [11, 12, 18]. Their results show that PKC is indeed feasible to be used in sensor networks. For example, Gura et al. [12] show that Elliptic Curve Cryptography (ECC) signature verification takes 1.62s with 160-bit keys on ATmega128 8MHz processor, a processor used for Crossbow motes platform [7].

Even with PKC getting faster and faster, performance difference between PKC and symmetric key cryptography is not going to change much unless some breakthrough in PKC occurs [8]. Compared to the symmetric key cryptography, the cost of PKC is still much more inefficient. For instance, a 64-bit RC5 encryption on ATmega128 8MHz takes 5.6 milliseconds, and a 160-bit SHA1 hash function evaluation takes only 7.2 milliseconds [10]. These are more than 200 times faster than PKC algorithms, and the gap is unlikely to significantly decrease. Furthermore, public key cryptography is not only computationally expensive, but also incurs communication overhead. For instance, to send a public key from one node to another using RSA, at least 1024bits needs to be sent if the private key is 1024 bits long [22]. Therefore, even after PKC is implemented in sensor nodes, we still need to treat PKC as expensive operations, and we need to use it more selectively and efficiently in order to maximize the lifetime of sensor networks.

Du et. al. developed a tree-based key-authentication scheme in order to minimize the PKC authentication overhead [8]. They modified the model of Merkle trees that was proposed by Merkle in 1980 [20]. The Merkle tree is basically a complete binary tree that has all sensors' public keys as its leaves. Then, every parent node in the Merkle tree is the result of a *hash* function of its left and right children. Therefore, there is a unique path between a sensor's public key and the root $(pk, h_{l-1}, h_{l-2}, \cdots, h_1)$, where pk is the sensor's public key, h_i is the hash value of path node of level i, and l is the tree height. Every sensor that wants to authenticate itself has to save the corresponding tree path in its memory. On the other hand, every sensor that wants to certify other sensors' keys needs to save the root of the Merkle tree only. After sensors deployment, a sensor should send its public key with the list of corresponding siblings of the path $(pk, h_{l-1}, h_{l-2}, \cdots, h_2)$, namely $(sibling(pk), sibling(h_{l-1}), sibling(h_{l-2}), \cdots, sibling(h_2))$, in order to authenticate its public key. The recipient sensor calculates $h'_{l-1} = hash(pk, sibling(pk))$, $h'_{l-2} = hash(h'_{l-1}, sibling(h_{l-1}))$, $\cdots$, $h'_1 = hash(h'_2, sibling(h_2))$, and then verifies that h'_1 is identical to the stored Merkle tree root h_1. The Merkle tree technique relies on the fact that the number of calculated hash functions at the recipient node is $O(\log(\text{number of sensors}))$, which is still more efficient than a single public key encryption.

Du et. al. exploited the sensors deployment knowledge to reduce the height of the Merkle tree. In fact, they suggested the use of multiple trees, where shorter trees gather adjacent sensors' public keys, and larger trees are a collection of shorter sub-trees. The large trees serve to authenticate sensors that are not adjacent; the farther the communicating sensors, the larger tree they belong to. The main disadvantage of the modified Merkle scheme is the network scalability. In fact, adding a single node will affect the hash value at the root, and therefore, the entire network needs to be notified, wasting communication bandwidth and transmission power.

5 Symmetric Key Based Security

Even though the public key based authentication solves the problematic key-distribution process and secures sensor scalability, Eschenauer and Gligor argued against it [9] due to its tremendous computational overhead. Instead, they proposed a *basic-scheme* for random symmetric key pre-distribution [9]. They suggest that a pool of S keys is generated by a central authority, where only m keys are to be chosen (randomly) by each sensor. When the sensors are deployed, each node consults with its neighbors for a possible shared key among the m key stored in its memory. Eschenauer and Gligor assert that two sensors could share a key with probability p. If we consider a graph of all nodes where each edge represents a shared key between two adjacent sensors, the value of p is chosen so that the graph is connected; there exists a path between any two nodes in the network. Therefore, if two adjacent sensors do not share a key, a secret key is generated and transmitted through the connecting secure path. However, it is still unobvious how to choose the optimal values of S, m, and p. When m is small, adjacent sensors are not likely to share secret keys, and therefore, additional key regenerations are needed which reduces the sensor lifetime due to the additional transmission and computational overhead. On the other hand, a larger m exposes the network to fast compromise when a node is captured, in addition to the large space needed to store the keys.

In order to increase the resilience against node's cap-

ture, Chan et. al. [6] upgraded the above basic-scheme to a *q-composite scheme* where q common keys are needed in order to establish a secure link. When a sensor discovers that it shares q keys with its neighbor $(K_1, K_2, \cdots, K_q)$, a secret key is generated as a hash function of those shared keys; $K = hash(K_1, K_2, \cdots, K_q)$. So, the q-composite scheme is simply a basic-scheme with a larger pool of keys $S^{'} = \{hash(K_1, K_2, \cdots, K_q), (K_1, K_2, \cdots, K_q)S^q\}$. However, the q-composite scheme encounters the computational overhead of the hash function after sensors deployment, which consumes more energy.

6 Towards Dynamic Security: Efficiency of Stream Ciphers

Karlof et. al. designed a chain-block-cipher (CBC) security mechanism called TinySec [15], and argued why CBC is the most appropriate encryption scheme for sensor networks. Symmetric key encryption schemes generally fall into two categories: stream ciphers and modes of operation using block ciphers. The fastest stream ciphers are faster than the fastest block ciphers [27], which might make them more appealing in a resource-constrained environment. One kind of stream ciphers is the initialization vector (IV) based stream cipher, where both the secret key K and IV are used as a seed for pseudorandom encryption keystream generation, $G_K(IV)$. The keystream is then XORed against the plain message P: $C = (IV; G_K(IV) \bigoplus P)$. However, IV-based stream ciphers have a devastating failure mode: if the same IV is ever used to encrypt two different packets, then it is often possible to recover both plaintexts. So, another alternative is to use a mode of operation based on a block cipher [4].

A block cipher is a keyed pseudorandom permutation over small bit strings, typically 8 or 16 bytes. Examples of block ciphers include DES, AES, RC5, and Skipjack. Since we usually want to encrypt and authenticate messages longer than 8 or 16 bytes, block ciphers require a mode of operation to encrypt longer messages. For a k byte block cipher, a mode of operation typically divides a message into segments of k bytes blocks and uses the block cipher in a special way to encrypt the message block by block. Using a block cipher for encryption has an additional advantage. Since the most known message authentication code (MAC) algorithms use a block cipher [1], thus it should be available in the sensors software, conserving memory space. A mode of operation has to be used in block ciphers; the counter (CTR) mode would be one choice [1]. However, the CTR mode is a stream cipher mode of operation, and shares all the problems as the IV-based stream cipher. Another possible choice is cipher block chaining (CBC) mode [1]. CBC mode is provably secure when IVs do not repeat [1]. On the other hand, CBC leaks only a small amount of information in the presence of repeated IVs, a significant improvement over a IV-based stream cipher [15]. It is known to suffer some leakage when used with counter generated IVs, yet it is best used with random generation of IVs [15].

The use of the CBC mode with blocks of size 8-byte results in ciphertexts which are multiples of 8 bytes. Such choice of block size forces message expansion, which increases power consumption. Karlof et. al. [23] proposed the use of ciphertext stealing technique to ensure that the ciphertext is the same length as the underlying plaintext. Encrypting data payloads of less than 8 bytes will produce a ciphertext of 8 bytes because ciphertext stealing requires at least one block of ciphertext. However, ciphertext stealing requires additional computation, which degrades the sensors power lifetime.

Another major issue in network security is cipher integrity. History has proven that using encryption without message authentication is insecure [3, 5, 16]. For example, flipping bits in unauthenticated encrypted messages can cause predictable changes in the plaintext [5]. Yet, without message authentication mechanism to guarantee integrity, receivers will not be able to detect such changes. Unauthenticated messages are also vulnerable to cut-and-paste attacks [3]. In a cut-and-paste attack, an adversary breaks apart an unauthenticated encrypted message and constructs another message's cipher, which may decrypt to something meaningful, escaping detection. For example, if all the authorized nodes share a single key, an adversary can extract the encrypted data payload from a message to one node and send it to different node. Since the encrypted payload is unaltered, the second node will successfully decrypt and accept the message.

TinySec uses a cipher block chaining construction, CBC-MAC [2], for computing and verifying MACs. CBC-MAC is efficient and fast, and the fact that it relies on a block cipher as well minimizes the number of implemented cryptographic primitives in the limited size sensor memory. CBC-MAC is provably secure [2], however the standard CBC-MAC construction is not secure for variably sized messages. Bellare, Kilian, and Rogaway suggest three alternatives for generating MACs for variable sized messages [2].

7 Synchronous Dynamic Encryption System

Due to the restrictions imposed on sensors networks, our major objective in designing a new security model is to minimize cost-effect of the following:

1) Network intrusion, when a sensor is captured.

2) Communication overhead, in case of revoking a shared secret key, especially the keys stored at the captured sensor.

3) Computation overhead, in securing the network, in order to save sensor's lifetime.

4) Utilized key space.

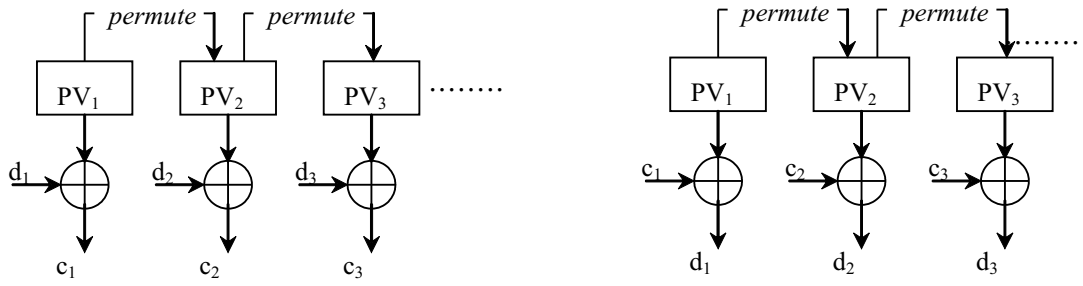

Figure 1: (a) Encryption (b) Decryption

Throughout the rest of this paper, we will focus on the design of our security model following the first, second, and third objective. The key establishment method is adopted from other peer mechanisms.

Our Synchronous Dynamic Encryption System (SDES) is a stream cipher crypto system based on permutation vector generation. SDES that avoids the usage of an IV, due to its security loopholes, previously mentioned. Upon key agreement, the two communicating sensors use the shared key K in any future invocation of security primitives, e.g., communication confidentiality, sensor authentication, and message integrity. We will adopt the basic-scheme of random key pre-distribution proposed by Eschenauer and Gligor [9], since key agreement methods are out of the scope of this paper. Yet, we will also utilize the PKC in the case the sensor fails to authenticate itself to the network. For instance, when a new sensor joins the network, it is preferable to perform a certification of its public key rather than jeopardizing the entire network security using the symmetric key approach. We will elaborate further on the authentication process, later in the paper.

The design strategy of our security mechanism is based on the following three concepts. Firstly, the SDES has a total separation of its two major processes, namely re-keying and data encryption. In order to achieve such separation, we are using two different keys for the aforementioned processes. Hence, with the very remote possibility of breaking the encryption key, an intruder will not be able to have any shot at the more important secret key, unless brute-forcing it. Secondly, for time efficiency, both processes are very simple and flat for speedy processing. Thirdly, the ingredient involved in the manufacturing of the secret key must produce a fail safe key against most, if not all, known security attacks.

SDES is a stream cipher that generates a stream of permutation vectors. Each permutation vector of size n is combination of byte cells, each with value that ranges between 0 and $n-1$, without repetition. For instance, (0, 1, 2), (1, 0, 2), and (2, 1, 0) are permutation vectors of size 3. These permutation vectors are used to encrypt plain messages, one at a time. In order to simplify the key management procedure, SDES uses the secret key K in the key management process in order to permute the current encryption key, permutation vector, to produce the next permutation vector or encryption.

The generation of encrypting permutation vectors keys can be performed recursively. Given a permutation vector PV, the generation of the next permutation vector $PV^{`} = Permutation(PV, K)$ is performed as follows:

$$\text{for } j \longleftarrow 1 \text{ to } n \text{ do}$$
$$\text{swap } (PV[j], PV[K[j] \text{ modulo } [n]).$$

Given the initial permutation vector is $(0, 1, \cdots, n-1)$, each node can generate a list of permutation vector that can be used in future encryption. This is possible only if the secret key K is static. However, for security purposes, we prefer that the secret key will be modified as the encryption key is regenerated. We propose also that K is generated based on the communication history. Assuming that communication data between two specific sensors is independent of any other communications that occur on the network, sensor's capture will have as minimal affect on the network security as possible. In fact, the only compromised communications are those involving the violated node. Therefore, SDES achieves the same security level of the PKC in terms of alleviating the node's capture effect.

7.1 Encryption/Decryption and Re-keying Functions

The encryption function is simplified in order to minimize the computational overhead, saving sensor's battery power. Thus, a simple XOR is performed between the data record d_i and a generated permutation vectors PV_i, resulting in a cipher c_i, to be transmitted, Figure 1a. The decryption function, at the receiving sensor, is performed in the same manner of the encryption function, at the source sensor. The cipher record c_i is XORed with the same permutation vector PV_i (generated at the recipient side) producing the original data record d_i, Figure 1b. As part of the full source-destination secret key synchronization (hence the name SDES), both communication parties generate the same new permutation vector (PV_{i+1}), based on the same previously generated secrete K, to be used in the next encryption/decryption operations.

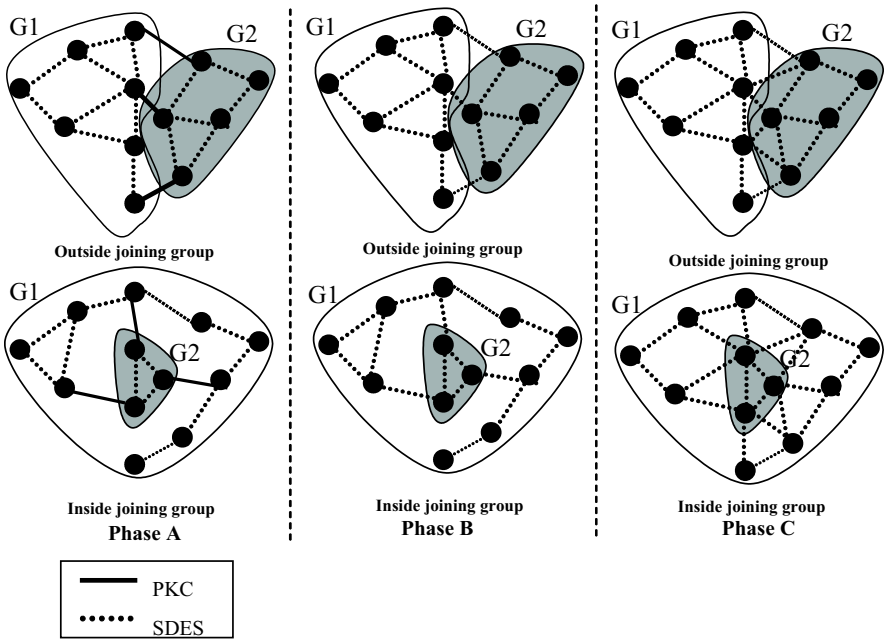

Figure 2: Authentication of joining group of sensors: -Phase A: PCK authentication of every edge sensor in G2 with one edge sensor from G1, and SDES authentication between all adjacent sensors in G2. -Phase B: SDES key establishment between already PCK authenticated sensors from G2 and their corresponding authenticating sensors from G1. -Phase C: SDES key establishment between edge sensors G2 and any other adjacent sensor from G1.

The next step is the source-destination synchronized re-keying process. As we discussed earlier, SDES alters the secret key for solid network security. Our SDES is novel; it distinguishes itself from other peer techniques by involving both the encryption key PV_i and the data d_i in the secret key re-keying function as follows: $K_{i+1} \longleftarrow (K_i + PV_i + d_i)$ modulo n, where K_i and K_{i+1} are the respective current and the next secret key values.

There are three major advantages of our re-keying approach. Firstly, since PV_i is very diverse, the generated secret key will be robust, avoiding poor-key attacks. Moreover, the overall network security is increased due the natural diversity of transmitted data between different communication links. In order to verify the equi-distribution of the generated secret key bytes, we carried out 10^7 encryption operations [25], with both random and constant data records. Results showed an equi-distribution of the byte value repetition with an average of 39062.5, and a standard deviation of 0.5%, which confirm that our re-kying mechanism produces non-biased keys.

The second advantage of SDES re-keying over other peer mechanisms is that all past communicated information is safe, prior to compromising any secret key instance. Assuming that an attacker succeeds in capturing a sensor at the i^{th} communication phase, he/she will compromise any data that is stored in the sensor's memory. Therefore, all secret information K_i, PV_i and d_i are compromised, in addition to any cipher $(c_1, c_2, \cdots, c_i)$ that the attacker may record while eavesdropping on the communication channel. Since the sensor may share the same set of initial m keys that was described in the basic-scheme of random key distribution, the attacker needs to perform backward tracking to get the initial key K_1, which is so critical since it is the only key that could be temporarily shared by a group of sensors. Because the secret information at the $(i-1)^{th}$ phase $(K_{i-1}, PV_{i-1} and d_{i-1})$ is overwritten, the attacker has to solve these equations in order to restore them back:

- $K_{i-1} = (K_i - PV_{i-1} - d_{i-1})$ modulo n.
- $PV_{i-1} = ReversePermutation(PV_i, K_{i-1})$
- $d_{i-1} = (K_i - K_{i-1} - PV_{i-1})$ modulo n, or $d_{i-1} = c_{i-1} \oplus K_{i-1}$

where *ReversePermutation()* is the reverse function of *Permutation()*. However, any attempt to solve the previous equations via variable substitution process will lead to circular referencing; i.e., K_{i-1}, PV_{i-1}, and d_{i-1} in function of K_{i-1}, PV_{i-1}, and d_{i-1}, respectively. Moreover, we have proved mathematically (out of the scope of this paper) that at least 2^{n-1} different values for K_{i-1} will permute PV_{i-1} to PV_i. Therefore, the backward tracking to get up to K_1 is as hard as brute-forcing a key space of $\Omega(2^{n-1})$ size complexity. For considerably large values of n, the proposed model guarantees minimal security damage upon sensor capturing; jeopardizing the corresponding communication links of the captured node. However, the proposed model still relies on the fact that sensors are not likely to be captured at the beginning of the network deployment. In case of such an early capture, our model

Table 1: The correspondent violation rates versus fixed round sizes in order to reach 99.99% gross protocol efficiency

Round Size (R)	**Violation Rate of the Hostile Environment**(p)	**Expected Useful Protocol Efficiency**(UPE)
5	$0.055912 - 2.000110^{-05}$	$60\% - 80\%$
10	$0.028358 - 1.0000510^{-05}$	$67.5\% - 90\%$
20	$0.014281 - 5.0002410^{-06}$	$71.25\% - 95\%$
50	$0.005737 - 2.000110^{-06}$	$73.5\% - 98\%$
100	$0.002873 - 1.0000510^{-06}$	$74.25\% - 99\%$

inherits the same vulnerabilities of the basic-scheme of random key pre-distribution.

The third advantage of our re-keying method is the reduction of key revocation effect. When a malicious node is detected, other peer mechanisms need to broadcast the entire set of keys corresponding to the malicious node. They also encounter an additional overhead of key re-establishment for all revoked keys. In our system, only the malicious node id is to be broadcasted to the entire network in order to start the process of its isolation.

7.2 Sensors Authentication

After establishing a secret key based on the basic-scheme, any two communicating sensors have to start a mutual authentication procedure as follows. The source sensor sends a connection request to the destination with two challenges c_1 and c_2. The challenges quantities c_1 and c_2 are the encryption of the same randomly generated nonce N with two consecutive permutation vectors, which are generated based on the shared key K. The authentication process starts by obtaining the decrypted values of c_1 and c_2 and comparing them for equality. If they are equal, the recipient node encrypts the same nonce with the next permutation vector, in sequence, and sends the resulting cipher c_3 back to the source sensor. The mutual authentication ends successfully when the source sensor decrypts c_3 and verifies the equality with the originally transmitted N.

New sensors deployment might be due to either the need to expand an existing network (say group G_1) to cover adjacent regions, or recover from a bad distribution of sensors in a previous deployment. Other peer mechanisms consider only individual node deployment, since a newly joining node might share several keys with the network. However, it is no the case in our system, since all keys are dynamic. Therefore, we implemented the deployment of new sensors in groups instead of individuals as shown in Figure 2.

We will follow the same basic-scheme key agreement and the above SDES authentication mechanisms in order to establish secure connection between sensors of the newly joining group G_2. In order to join G_1 and G_2, their touching edge sensors authenticate via the PKC, only once in their lifetime, saving much needed sensor energy. Then, based on the basic-scheme in Section V, the process of neighborhood adjustment with secret key establishment will be accomplished among adjacent sensors of G_1 and G_2.

7.3 Additional Efficiency Through Flexible Integrity Checking Process

The SDES introduces a much more flexible and powerful message integrity mechanism replacing the checksum-like mechanisms. Since secret key generation depends on data, integrity violations are detected via key mis-synchronization. Our system divides each session into sections of size $R + 1$ records, with R data records and a duplicate of the last record. The receiver validates the integrity of the entire section by simply verifying the equality of the last two received data records, ignoring the duplicate record. If an integrity violation is detected, the sender needs to encrypt and retransmit the previous $R+1$ data records, which may degrade the performance of our mechanism in hostile environments. An advantage of our mechanism is the flexibility of adjusting R based on the environment hostility.

Hence, the communication throughput is $R/(R + 1)$, whereas the efficiency of other mechanisms that utilize CRC is about 80% (e.g., in 128-block ciphers, using 32-bit CRC field). Moreover, the CRC detects only cipher alteration, whereas our mechanism has the huge advantage of detecting many more violations, such as cipher shuffling, injection, and deletion and session hijacking.

Network Efficiency Of CBC-MAC Integrity Mechanism

Next, we will analyze the effective throughput of securely transmitting N bits using rigid mechanisms, like CBC-MAC. Let PS be the packet size in bits, and CMS the corresponding CBC-MAC filed size. Therefore, the total number of packets is N/PS, and the number of non-violated packets ranges between 0 and N/PS.

Let us recall that $(1-p)$ is the probability that a packet is not violated. Therefore, $\frac{n!}{i!(n-1)!}(1-p)^i P^{n-i})$(where $n = N/PS$) is the probability of i non-violated packets, since they can be in any order. Then, the expected num-

Table 2: Comparison of deployed security mechanisms with SDES (n is network size, m is the maximum keys that can be stored in a sensor's memory)

Technique	Key storage per sensor	Scalability	Affected nodes due to a sensor intrusion	Sensor revocation overhead
PCK	2 (a key and a certificate)	node-to-node authentication	only sensors communicating with the intruded node	Broadcast intruded node id
Merkle Tree	$1 + \text{Log } n$	complex tree modification	only sensors communicating with the intruded node	Broadcast intruded node id
Basic-scheme of Random Key	m	Simple with probability p	all sensors that share the same set of m keys stored at the intruded node	Broadcast intruded node id and its set of m secret keys
SDES	m	PCK-like for touching edge sensors, and basic-scheme-like for others.	only sensors communicating with the intruded node	Broadcast intruded node id

ber of non-violated packets is:

$$E = \sum_{i=0}^{n} \frac{n!}{i!(n-1)!}(1-p)^i P^{n-i}.$$

This expectation is identical to that of the Bernouli distribution, which has the mean of $n \times (1-p)$. Therefore, the expected *useful* throughput of transmitting N bits is:

$$\begin{aligned}[PS-CMS] \times E &= [PS-CMS] \times n \times (1-p) \\ &= [PS-CMS] \times N/PS \times (1-p) \\ &= \frac{PS-CMS}{PS}(1-p)N\end{aligned}$$

which means that the useful protocol efficiency using CRC is:

$$\frac{PS-CMS}{PS}(1-p). \quad (1)$$

Network Efficiency of SDES Integrity Mechanism

Let Q be the probability to transmit R records with no integrity violation, i.e., $Q = (1-p)^R$. Then, the probability to transmit R records with at least one integrity violation (of one record) is $P = 1-Q = 1-(1-p)^R$. Let X be the random variable that represents the total number of transmitted records in order to get R records transmitted successfully. Therefore, $Pr(X = R) = (1-P), Pr(X = 2R) = (1-P)P, Pr(X = 3R) = (1-P)P^2, \cdots, Pr(X = nR) = (1-P)P^n$.

Then, the expected value of X is $E(X) = \sum_{n=1}^{\infty} nRPr(nR)$, therefore, $E(X) = \sum_{n=1}^{\infty} nR(1-P)P^{n-1}$, then, $E(X) = R(1-P)\sum_{n=1}^{\infty} nP^{n-1}$, therefore, $E(X) = R(1-P)\frac{d[\sum_{n=1}^{\infty} P^n]}{dP}$, so, $E(X) = R(1-P)\frac{d[\frac{P}{1-P}}{dP}$, then, $E(X) = R(1-P)\frac{1}{(1-P)^2}$, thus, $E(X) = \frac{R}{1-P} = \frac{R}{(1-p)^R}$.

Since $E(X)$ is the average number of transmitted records in order to get R records transmitted successfully, the gross protocol efficiency GPE is:

$$gross\ protocol\ efficiency = \frac{R}{E(X)} = (1-p)^R. \quad (2)$$

However, the useful protocol efficiency is less than the gross protocol efficiency, since the significant payload consists of $R-1$ records only. Therefore, the useful protocol efficiency UPE is:

$$useful\ protocol\ efficiency = \frac{R-1}{R}\ gross\ protocol\ efficiency. \quad (3)$$

So, given the integrity violation probability p and a desired GPE, the recommended value of R is: $R = \frac{\ln(GPE)}{\ln(1-p)}$.

Despite the fact that a 99.99% GPE is achievable with reasonable violation probabilities, network users are concerned more about the net network efficiency (UPE). As shown in Table 1, It is quiet high to achieve high UPE with small values of R, even with small p. Therefore, the network should tune the round size to lower values as it detects higher rates of violations.

Upon detecting records violations, in addition to approximating p, a calculated adjustment to R is in order, for achieving the highest useful protocol throughput.

We have: $UPE = \frac{R-1}{R}(1-p)^R$ based on Equations (2) and (3), therefore, $\ln(UPE) = \ln[\frac{R-1}{R}] + R\ln(1-p)$. In order to solve the previous equation, we need to use the next Taylor expansion: $[\frac{x+1}{x}] = 2[\frac{1}{2x+1} + o[\frac{1}{2x+1}]]$, where $o[\frac{1}{2x+1}]$ is an insignificant term compared to $\frac{1}{2x+1}$. Therefore, after Taylor expansion substitution, the previous equation is transformed to: $\ln(UPE) = \ln\frac{-2}{2R-1} + R\ln(1-$

Table 3: Encryption/Decryption power consumption of 256×10^4 bytes using $8M_{HZ}$ CPU

Encryption Method	**Encryption Power Consumption (power unit)**	**Decryption Power Consumption (power unit)**
SDES	2.1	2.206
AES	3.5876	3.6188
AES (CBC Mode)	4.0188	4.256
AES (CBC-MAC)	4.0175	4.137
DES	10.5688	11.2
DES (CBC Mode)	11.1812	13.1688
Triple DES	33.0372	30.8628
Triple DES (CBC Mode)	39.256	31.5876

p). For clarity purposes, we will substitute $\ln(UPE)$ with a, and $\ln(1-p)$ with b, ending up with $a = \frac{-2}{2R-1} + Rb$, which is equivalent to the quadratic equation:

$$2bR^2 - (2a+b)R + a - 2 = 0 \tag{4}$$

which has the discriminant $\Delta = 4a^2 - 4ab + 16b + b^2$.

In order to solve Equation (4), we need to determine the roots values of a or b that make Δ positive. Consequently, we need to solve $\Delta = 0$ with respect to a. Then, the corresponding discriminant is $\Delta_a = -256b$, which is always positive, since $1-p$ is less than one, and then b renders negative. So, the quadratic equation $\Delta = 0$, with respect to a, has two roots:

$$a_{1,2} = \frac{b \pm 4\sqrt{-b}}{2}. \tag{5}$$

Hence, when $a \leq a_2$, Δ is positive, which set a_2 to be the maximum achievable value for a. Note that $a \geq a_1$ also renders Δ positive, but this case is discarded, since a_1 is positive; a is equal to $\ln(NNE)$ which is always negative. Hence, Equation (4) has two roots:

$$R_{1,2} = \frac{2a + b \pm \sqrt{4a^2 - 4ab + 16b + b^2}}{4b}. \tag{6}$$

So, given a violation probability p, the corresponding maximum achievable UPE is $e^{a_2} = e^{\frac{\ln(1-p) - 4\sqrt{-\ln(1-p)}}{2}}$, based on Equation (5). Consequently, based on Equation (6), we considered the value of R_1 in order to achieve such efficiency (R_2 is identical to R_1 when $a = a_2$, which makes $\Delta = 0$):

$$R_{1|a=a_2} = R_{\text{Maximum } UPE} = \frac{1}{2} - \frac{\sqrt{-\ln(1-p)}}{\ln(1-p)}. \tag{7}$$

7.4 Comparison with Peer Techniques

Table 2 summarizes the major advantages and disadvantages of the previously discussed security mechanisms compared with our proposed model SDES.

In terms of power consumption, our SDES mechanism is the lowest among all due to the simplicity of stream ciphers, as shown in the next section. Moreover, the flexibility of SDES saves more energy, when the need for integrity protection rises, than other listed techniques. In terms of memory storage, SDES follows the same mechanism of the basic-scheme of random keys, utilizing only the available space of the sensor's memory.

Scalability is also a very important factor in sensor networks since there will be always a need to replace expired/damaged sensors, increase the network diameter, populate sparsely distributed regions in the network. PCK techniques excel in the simplicity of their scalability, which led us to incorporate their approach temporarily in our system. Since our mechanism adopt a group sensor deployment, only edge nodes need to be PCK authenticated to the network. However, for other nodes, we use the simpler basic-scheme establishment. Hence, our hybrid scalability is more effective than that of the PCK in terms of computation and communication overheads, which saves energy. Another advantage of SDES's scalability over that of the basic-scheme is the balancing of power consumption over the network. In the latter, a key path has to be established through the network if a new joining sensor does not share keys with its neighbors. This could lead to imbalanced power consumption at nodes that are part of multi-paths. On the contrary, the SDES scalability guarantees the path key establishment only within the newly joining group, decreasing the possibility of multiple shared paths per node, hence alleviating the above problem.

The impact of capturing (intruding) a node varies from one security mechanism to another. Since our SDES mechanism applies re-keying to all sensors stored keys upon the first communication, any sensor capture will jeopardize its corresponding communication links only. Hence, our SDES mechanism achieves the same best security of the PCK.

In case of detecting an intruded sensor, an efficient method is needed to isolate it from the network. Our mechanism saves communication bandwidth and energy since only the malicious node's id is broadcasted. Other mechanisms, such as the basic-scheme, need to broadcast the list of all keys that are stored at the malicious node.

Table 4: Energy consumption comparison between SDES and AES with CBC-MAC in a hostile environment.

Probability Violation	AES with CBC-MAC (PS=160 bits, CMS=32bits)		SDES		
	UPE	Energy Consumption (power unit)	UPE	R	Energy Consumption (power unit)
10^{-05}	80 %	10.04	99.37 %	317	2.11
10^{-04}	80 %	10.04	98.01 %	100	2.14
0.001	80 %	10.05	93.82 %	32	2.24
0.005	80 %	10.09	86.58 %	15	2.43
0.01	79 %	10.14	81.42 %	10	2.58
0.05	76 %	10.57	61.96 %	5	3.39
0.1	72 %	11.15	49.57 %	4	4.24
0.2	64 %	12.55	34.77 %	3	6.04
0.3	56 %	14.34	25.34 %	2	8.29
0.4	48 %	16.73	18.55 %	2	11.32
0.5	40 %	20.08	13.38 %	2	15.70
0.6	32 %	25.09	9.32 %	2	22.52
0.7	24 %	33.49	6.10 %	1	34.41
0.8	16 %	50.19	3.54 %	1	59.39
0.9	8 %	100.38	1.52 %	1	138.11
0.99	8 %	1003.75	1.37 %	1	1535.26

Notes: This energy consumption is relative to transmitting useful 256×10^4 bytes successfully.

Consequently, the revoked keys need to be re-established which adds another burden on the bandwidth and the sensor's power consumption.

8 Experiments and Analysis

In order to evaluate our system, we carried out several experiments with respect to encryption/ decryption power consumption. The experiments also involved the AES, DES, and Triple DES with both regular and CBC mode. Pottie and Kaiser [21] listed that the energy consumed in transmitting a 1K-bit packet over 100m is approximately the same as performing 3 million instructions on a typical scenario. Thus, with an 8MHz CPU, the energy spent on running CPU for 1 millisecond is equivalent to sending 2.67 bits. In the following analysis, we will consider the power unit as the power spent to run the sensor's CPU for 1 millisecond. The depicted results were obtained as the average of 50 runs, on the same benchmark.

Table 3 clearly shows the power saving advantage of SDES compared to other symmetric key techniques. Moreover, when the CBC mode is deployed to protect against replay attacks, the peer symmetric key techniques waste more energy due to the extra XOR operation of the CBC mode. For instance, when AES-CBC is deployed for data encryption, it will still consume energy equivalent to 200% of that used by our mechanism. In addition, AES-CBC-MAC spends approximately another 200% more energy than SDES in order to protect against data integrity violations. Hence, the simplicity and the flexibility of our mechanism led to 75% saving of sensor's energy compared to AES-based techniques (e.g., TinySec), an achievement that makes our security system the most amenable for wireless sensor networks.

Table 4 compares the maximum SDES UPE with that of the AES with CBC-MAC as a function of the violation probability (p), based on Equations 1 and 7, and Table 3. Results show that SDES achieves better UPE than AES-CBC-MAC, when p is less than 1%. Moreover, in terms of energy consumption, SDES has uses less power in highly hostile environment; saving between 12 % and 80 % for $p = 0.60$ down to 0, respectively. Notice that Equations (1) and (7) do not include the encryption overhead. Based on Pottie and Kaiser formula and Table 3, the ratio of encryption overhead over transmission overhead is $13.03 \times 10^{-8} (i.e., 2.67 bits/(256 \times 10^4 \times 8 bits))$, which is negligible.

9 Conclusion

In this paper, we introduced a novel security mechanism that is most amenable for deployment in the wireless sensor networks. The strength of our encryption technique, with a brute-force time complexity of $\Omega(2^n)$, is stemmed from a simple permutation style.

From existing key distribution systems, we adopted the random key pre-distribution technique for its efficiency in terms of secret keys space usage. Moreover, we adopted the PKC method for authenticating newly joining sensors. However, we proposed two simple joining schemes that reduce the power consumption of PKC utilization.

Novel to our technique, the initial keys stored in the sensor's memory are re-keyed with every performed en-

cryption/decryption, which limits the effect of sensor's intrusion to only its communication links. Moreover, key revocation overhead is reduced to simply broadcasting the intruded node id instead of its captured set of keys.

We also introduced a new integrity mechanism that adapts to the hostility of the network environment. Instead of encountering a rigid overhead for data integrity, as done in peer mechanisms (e.g., CRC, CBC-MAC), we developed a flexible mechanism that saves bandwidth and reduces sensor energy consumption. Such achievement is stemmed from a novel re-keying mechanism that involves data in addition to the encryption key.

Simulation results show that our mechanism has a clear set of advantages over existing peer mechanisms. SDES possesses better power budget and balancing, more immunity against sensor intrusion, in addition to man-in-the-middle and replay attacks.

References

[1] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, "A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation," in *Proceedings of 38th Annual Symposium on Foundations of Computer Science (FOCS 97)*, pp. 406-415, 1997.

[2] M. Bellare, J. Kilian, and P. Rogaway, "The security of the cipher block chaining message authentication code," *Journal of Computer and System Sciences*, vol. 61, no. 3, pp. 362-399, Dec. 2000.

[3] S. M. Bellovin. "Problem areas for the IP security protocols," in *Proceedings of the Sixth USENIX Security Symposium*, pp. 1-16, 1996.

[4] S. M. Bellovin and M. Blaze, "Cryptographic modes of operation for the Internet," in *Second NIST Workshop on Modes of Operation*, Aug. 2001.

[5] N. Borisov, I. Goldberg, and D. Wagner, "Intercepting mobile communications: The insecurity of 802.11," in *The Seventh Annual International Conference on Mobile Computing and Networking (MobiCom 2001)*, pp. 180-188, 2001.

[6] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 197-213, 2003.

[7] Crossbow Technology Inc., *Wireless Sensor Networks*, 2004, http://www.xbow.com/.

[8] W. Du, R. Wang, and P. Ning, "An efficient scheme for authenticating public keys in sensor networks," in *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pp. 58-67, May 2005.

[9] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pp. 41-47, Nov. 2002.

[10] P. Ganesan, R. Venugopalan, P. Peddabachagari, A. Dean, F. Mueller, and M. Sichitiu, "Analyzing and modeling encryption overhead for sensor network nodes," in *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pp. 151-159, San Diego, California, USA, Sept. 19, 2003.

[11] G. Gaubatz, J. Kaps, and B. Sunar, "Public keys cryptography in sensor networks - revisited," in *The Proceedings of the 1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS)*, pp. 2-18, 2004.

[12] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing elliptic curve cryptography and rSA on 8-bit CPUs," *CHES 04*, Aug. 11-13, 2004.

[13] Q. Huang, J. Cukier, H. Kobayashi, B. Liu, and J. Zhang, "Fast authenticated key establishment protocols for self-organizing sensor networks," in *WSNA'03*, pp. 141-150, San Diego, California, USA, Sept. 2003.

[14] IEEE Std. 802.15.4-2003, *IEEE Standard for Information Technology- Telecommunications and Information Exchange Between Systems-local and Metropolitan Area Networks - Specific Requirements - Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs)*, New York: IEEE Press, 2003.

[15] C. Karlof, N. Sastry, and D. Wagner, "TinySec: A link layer security architecture for wireless sensor networks," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pp. 162-175, Nov. 2004.

[16] H. Krawczyk, "The order of encryption and authentication for protecting communications (or: How secure is SSL?)," in *CRYPTO 2001*, LNCS 2139, pp. 310-331, Springer-Verlag, 2001.

[17] H. Luo, P. Zefros, J. Kong, S. Lu, and L. Zhang, "Self-securing Ad Hoc wireless networks," in *7th IEEE Symposium on Computers and Communications (ISCC '02)*, pp. 567-574, 2002.

[18] D. J. Malan, M. Welsh, and M. D. Smith, "A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography," in *The First IEEE International Conference on Sensor and Ad Hoc Communications and Networks*, Santa Clara, California, Oct. 2004.

[19] A. Menezes, P. V. Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Inc., 1997.

[20] R. Merkle, "Protocols for public key cryptosystems," in *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pp. 122-133, Apr. 1980.

[21] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Communications of the ACM*, pp. 51-58, May 2000.

[22] R. L. Rivest, A. Shamir, and L.A. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120-126, 1978.

[23] B. Schneier, *Applied Cryptography*, 2nd Edition, John Wiley & Sons, 1996.

[24] V. Shoup, "Practical threshold signatures," in *EUROCRYPT'00*, pp. 207-220, 2000.
[25] H. S. Soliman and M. Omari, "New design strategy of dynamic security implementation," *IEEE Workshop on Adaptive Wireless Networks (Globecom 2004)*, pp. 442-446, Dallas, TX, Dec. 3, 2004.
[26] F. Stajano and R. Anderson, "The resurrecting suckling: Security issues in Ad-Hoc wireless networks," in *7th International Workshop on Security Protocols*, LNCS 1796, pp. 172-194, Springer-Verlag, 1999.
[27] R. Venugopalan, P. Ganesan, P. Peddabachagari, A. Dean, F. Mueller, and M. Sichitiu, "Encryption overhead in embedded systems and sensor network nodes: Modeling and analysis," in *2003 International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, pp. 188-197, 2003.
[28] R. Wash, *Lecture Note on Stream Ciphers and RC4*, http://www.crimelabs.net/docs/stream.pdf.
[29] L. Zhou and Z. J. Haas, "Securing Ad Hoc networks," *IEEE Network Magazine*, vol. 13, no. 6, pp. 24-30, Nov./Dec. 1999.

Hamdy Soliman Received his B.S. in 1978, from Alexandria University (Egypt), and his M.S. in 1983 from Florida Institute of Technology, and his Ph.D. 1989 from New Mexico State University (USA), all degrees are in Computer Science. He is currently an Associate Professor in the computer science department at New Mexico Tech (NMT). His research interests include neural networks application in data/image compression, fiber networks admission control, computer and networks security, wireless security and protocols.

Mohammed Omari Received his B.E degree in computer science from the University of Es-Senia Oran, Algeria, in 1995 and the M.S. degree in computer science from New Mexico Tech, New Mexico, USA, in 2002, as well as the Ph.D. degree in 2005.

He is currently an Assistant Professor with the University of Adrar, Algeria. His research interests include network security, cryptography, sensors and ad hoc protocols, image processing, and neural networks.