TURKISH NAVAL ACADEMY
NAVAL SCIENCE AND ENGINEERING INSTITUTE
DEPARTMENT OF COMPUTER ENGINEERING

SECURITY IN WIRELESS SENSOR NETWORKS

MASTER THESIS

SERDAR SANCAK

Advisor : Assist.Prof. Vedat Coşkun
Co-Advisor : Dr. Erdal Çayırcı

İstanbul, 2003

SECURITY IN WIRELESS SENSOR NETWORKS

Submitted in partial fulfillment of the requirements for degree of

MASTER OF SCIENCE IN COMPUTER ENGINEERING

from the

TURKISH NAVAL ACADEMY

Author :

Serdar Sancak

Defense Date :30 / 07 / 2003
Approved by:

Assist.Prof. Vedat Coşkun (Advisor)

Dr. Erdal Çayırcı (Co-Advisor)

Prof. M.Yahya Karslıgil (Defense Committee Member)

Assist.Prof. Albert Levi (Defense Committee Member)

Assist.Prof. Gökhan Yavuz (Defense Committee Member)

ABSTRACT (TURKISH)

TELSİZ SENSÖR AĞLARINDA GÜVENLİK

Anahtar Kelimeler : Telsiz Sensör Ağları, Güvenlik, Sensör Savaşları, Spam Saldırıları, İstenmeyen Mesajlar, Karantina Bölgesi, Kimlik Denetimi, Pil Tüketimi.

Bir *sensör ağı*, telsiz iletişim yapısını kullanan ve kendi kendine organize olabilen çok sayıda sensörün bir araya gelerek birlikte çalışmasıyla oluşur. Sensör ağlarının, esnek yapıları, kendi kendilerine organize olabilmeleri ve düşük maliyetleri nedeniyle, özellikle savaş alanlarında pekçok uygulama olanağı bulunmaktadır. Bazı muhtemel askeri uygulamalar: dost kuvvetlerin, donanımın ve mühimmatın takip ve kontrolü; savaş alanının gözlenmesi; düşman kuvvetlerinin ve arazinin keşfi; hedef tespiti; hasar tespiti; nükleer, biyolojik ve kimyasal (NBC) taarruzunun tepiti ve keşfi.

Güvenlik hususları, özellikle düşman alanlarında kullanılan telsiz sensör ağlarında çok önemlidir. Eğer sensörlere ulaşılabilirse, sensörler düşman tarafından toplanabilir veya imha edilebilir. Eğer sensörler tehlikeli veya bir süreliğine ulaşılamayacak alanlara atılmışsa, bu sensör ağına düşman sensörler (anti-sensörler) atılabilir. Anti-sensörler sürekli sahte mesajlar üretebilirler. Bu sahte mesajlar, özellikle sink'e yakın sensörlerin enerjisinin tükenmesine ve gayri faal duruma geçmesine neden olurlar. Biz bu tür saldırılara spam saldırıları diyoruz.

Bu tezde, spam saldırılara karşı bir çözüm önerdik ve bu çözümlerin performansını simülasyon kullanarak değerlendirdik. Anti-sensörlerin hareketli

olduğu senaryosunu düşündük. Ayrıca şifreleme ve sürekli kimlik doğrulama yöntemlerini kullanmadık. Çünkü sensörlerin hesaplama kapasiteleri, güçleri ve hafızaları kısıtlıdır. Bunun yerine, karantina bölgesi adını verdiğimiz bölgeler tanımladık. Anti-sensörün bulunduğu alan karantina bölgesine alınır. Karantina bölgesinde bulunan bütün sensörler belli bir süre sadece kimlik-doğrulaması yapılmış mesajlar gönderebilirler.

ABSTRACT (ENGLISH)

SECURITY IN WIRELESS SENSOR NETWORKS

Keywords : Wireless Sensor Networks, Security, Sensor Wars, Spam Attacks, Unsolicited Messages, Quarantine Region, Authentication, Battery Exhaustion.

A *sensor network* is a collection of sheer number of sensor nodes that collaboratively work by using a multi-hop wireless communications architecture. Sensor networks have wide-range of applications especially in a battlefield because of their flexible, low cost, and self organizing features. Some feasible military applications are: monitoring friendly forces, equipment and ammunition; battle-field surveillance; reconnaissance of opposing forces and terrain; targeting; battle damage assessment; and nuclear, biological and chemical (NBC) attack detection and reconnaissance.

Security issues are of the key issues especially in wireless sensor networks often deployed beyond the enemy lines. When they are reachable, they can be collected and/or destroyed by the enemy. If they are deployed in a hazardous region or a region not accessible for a time period, anti-nodes can be deployed inside the sensor network. Anti-nodes can generate frequent dummy messages. These dummy messages may cause nodes especially close to the sink fail quicker due to energy depletion. We name these types of attacks spam attacks.

In this thesis, we explain our practical solutions for spam attacks and we also evaluate the performance of our schemes by simulation. We consider mobile anti-nodes scenarios. We also do not use encryption and continuous

authentications, because sensor nodes have limited computation, power and memory. Instead we determine the regions, named quarantine regions, that may hold suspicious nodes, and the nodes in those regions relay only the authenticated packets for a time period.

DISCLAIMER STATEMENT

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Turkish Navy, Naval Academy, and Naval Science and Engineering Institute.

DEDICATION

To my wife Sibel and my daughter Sezin for their support, love, and inspiration.

ACKNOWLEDGEMENT

I would like to thank my advisors Dr. Vedat Coşkun, Dr. Erdal Çayırcı and also Dr. Albert Levi for their helpful discussions and comments. I would also like to sincerely thank Vice Admiral Lütfü Sancar and Prof.Dr. Süleyman Özkaynak for the kindness and assistance they provided to obtain the required materials.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS, ACRONYMS, AND SYMBOLS

θ	Multiplication Factor
β	System Parameter
$\mathbf{d_{max}}$	Maximum Relay Distance
$\mathbf{d_q}$	The Distance Between the Borders of a Quarantine Region and the Anti-node
B	Block size (in bytes) of the input to the approved hash function
ipad	Inner pad; the byte x'36' repeated *B* times
opad	Outer pad; the byte x'5c' repeated *B* times
x'N'	Hexadecimal notation, where each symbol in the string '*N*' represents 4 binary bits
$\|\|$	Concatenation
$\oplus$	Exclusive-Or operation.
DADS	Detect and Defend Spam
DoS	Denial of Service
GPS	Global Positioning System
HMAC	Hash-Based Message Authentication Code
ID	Identification
IDS	Intrusion Detection System
MAC	Message Authentication Code
NBC	Nuclear, Biological and Chemical

I. INTRODUCTION

A. WIRELESS SENSOR NETWORKS

1. What is a Wireless Sensor Network?

A sensor network is a collection of sheer number of sensor nodes that collaboratively work by using a multi-hop wireless communications architecture. Figure 1 shows a sample wireless sensor network. In this figure, we can see the sensor field and scattered sensor nodes. Every sensor node in this field has capability to collect data and route back to sink. Sink has much more capabilities than the sensor nodes. Sink relays the coming messages to the end user via internet or satellite. There may be a few sinks in a sensor field.

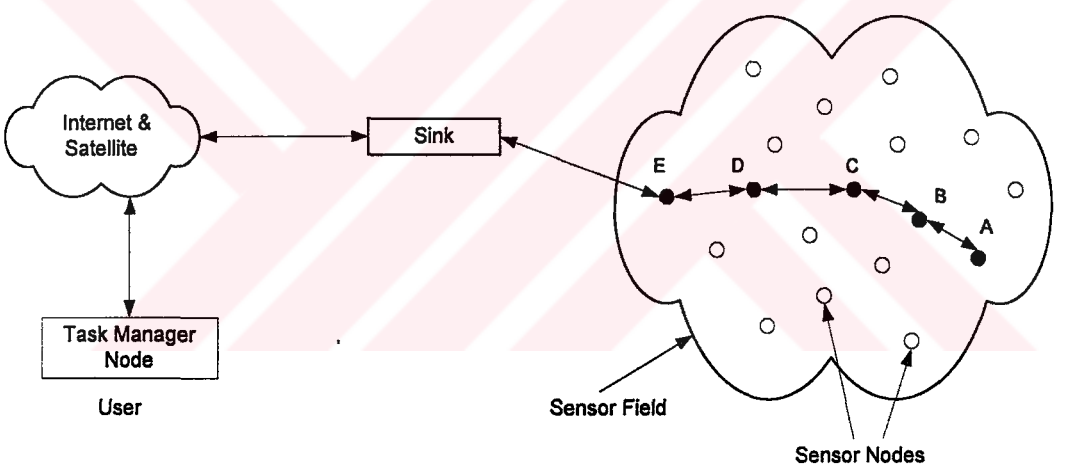

Figure 1. A simple wireless sensor network [1].

Important features of sensor nodes are outlined below:

- Their lifetime is generally limited with the lifetime of a tiny battery.
- They have limited computational power and memory.
- They are prone to failures.
- They are densely deployed. The distance between two nodes is often less then a few meters.

– Although their location is fixed in many applications, network topology changes frequently due to node failures and objects passing through the sensor field.

– Nodes are supposed to be location aware in many sensor network applications.

A sensor node is constituted four main components which are shown in Figure 2. These main components are: a sensing unit, a processing unit, a transceiver unit and a power unit. A sensor node may also has additional components such as location finding system, a power generator and a mobilizer according to the application [1].

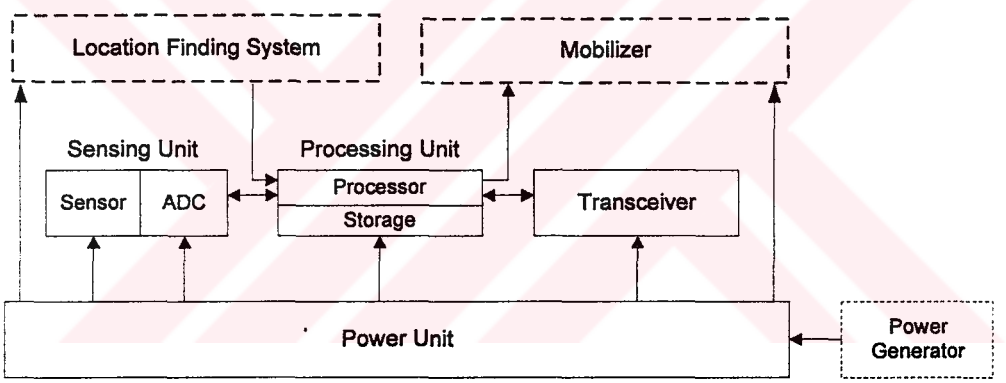

Figure 2. The components of a sensor node [1].

To show sensor nodes' hardware constraints, a Cots Dust Sensor Mote [2] is shown in Figure 3 and the mote's features are outlined below.

- 4Mhz, 8bit MCU (ATMEL)
- 512 bytes RAM, 8K ROM
- 900Mhz Radio (RF Monolithics)
- 10-100 ft. range
- Temperature Sensor

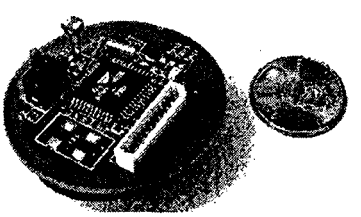

Figure 3. Cots Dust Sensor Mote [2]

- Light Sensor
- LED outputs
- Serial Port

2. Application Examples of Wireless Sensor Networks

Sensor networks may consist of seismic, low sampling rate magnetic, thermal, visual, infrared, acoustic and radar sensors. These different types of sensors may use to monitor the ambient conditions which include the following [1]:

- Temperature,
- Humidity,
- Vehicular Movement,
- Lightening condition,
- Pressure,
- Soil makeup,
- Noise levels,
- The presence or absence of certain kinds of objects,
- Mechanical stress levels on attached objects,
- The current characteristics such as speed, direction, and size of an object.

Sensor networks have a wide-range of applications especially in a battlefield because of their flexible, low cost, and self-organizing features. These applications are sketched in below [1].

a. Military Applications

- Monitoring friendly forces, equipment and ammunition,
- Battle-field surveillance,
- Reconnaissance of opposing forces and terrain,

- Targeting,
- Battle damage assessment, and
- Nuclear, biological and chemical (NBC) attack detection and reconnaissance.

b. Environmental Applications

- Forest fire detection,
- Flood detection,
- Biocomplexity mapping of the environment, and
- Precision agriculture.

c. Health Applications

- Telemonitoring of human physiological data,
- Tracking and monitoring doctors and patients inside a hospital,
- Drug administration in hospitals.

d. Home Applications

- Home automation, and
- Smart environment.

e. Other Commercial Applications

- Environmental control in office buildings,
- Interactive museums,
- Detecting and monitoring car thefts,
- Managing inventory control,
- Vehicle tracking and detection.

B. GOALS OF THIS THESIS

Security is one of the key issues especially in tactical wireless sensor networks often deployed beyond the enemy lines. When a sensor network is

reachable, sensor nodes can be collected or destroyed by the enemy. If they are deployed in a hazardous region or in a region not accessible for a period of time, hostile nodes, i.e., anti-nodes, can be deployed inside the sensor network. Anti-nodes can generate frequent dummy messages, i.e., unsolicited messages. These unsolicited messages may cause nodes especially close to the sink to fail sooner due to energy depletion. We call these types of attacks as spam attacks. To the best of our knowledge, there is not any security scheme that both considers deficiencies of tiny sensor nodes and prevails spam attacks in sensor networks.

In this thesis, we propose detect and defend spam (DADS) scheme for defending against spam attacks. DADS is based on message authentication in the vicinity of an anti-node. The area where authentication is required for a temporary period of time is called a quarantine region. Since the range of a quarantine region is limited and it is temporary, our scheme does not incur an excessive overhead for security. In this paper, we propose practical solutions for the following issues:

- How to detect anti-nodes.
- How to determine quarantine region boundaries.
- How to inform a node that it should relay only authenticated messages.
- How to authenticate.
- How to cancel a quarantine region.

C. STRUCTURE OF THIS THESIS

This thesis is organized as follows: Section II gives background information on cryptography, authentication and intrusion detection systems. Then explain security requirements, security vulnerabilities and security solutions of sensor networks.

In Section III, we introduce sensor wars and spam attacks, then explain our counter-measure technique Detect And Defend Spam (DADS) for spam attacks.

In Section IV, the performance of DADS scheme is evaluated. The scheme is discussed in the context of sensor density, number of anti-nodes, spam frequency and the size of quarantine region.

Finally, Section V gives concluding remarks of this thesis.

II. SECURITY IN WIRELESS SENSOR NETWORKS

A. BACKGROUND INFORMATION

1. Differences Between Adhoc and Sensor Networks

Differences between Adhoc and sensor networks are outlined below [1]:

- The number of sensor nodes in a sensor network can be several orders of magnitude higher than the nodes in an ad hoc network.
- Sensor nodes are densely deployed.
- Sensor nodes are prone to failures.
- The topology of a sensor network changes very frequently.
- Sensor nodes mainly use broadcast communication paradigm whereas most ad hoc networks are based on point-to-point communications.
- Sensor nodes are limited in power, computational capacities, and memory.
- Sensor nodes may not have global identification (ID) because of the large amount of overhead and large number of sensors.

2. Cryptography

We want to define some terms which are used in cryptography. An original message is known as *plaintext*. The process of hiding a plaintext's substance is *encryption*. Encrypted message is called as *ciphertext*. The process of reverting to plaintext from *ciphertext* is *decryption*. *Key* is an input to encryption algorithm and it is not related with *plaintext*.

Cryptography is described in [3] as "Cryptography is the study of 'mathematical' systems involving two kinds of security problems: privacy and authentication".

There are many cryptographic algorithms. But modern cryptography consists of two main key-based algorithms: 'private key cryptography' and 'public key cryptography'. But we do not give information about public key cryptography [3, 4, 5, 6]. Because it requires too much calculation, however sensor nodes' computational power and memory is limited. So we believe that it is beyond our scope. We also give information about 'hash functions' which is an important cryptographic primitive in security protocols.

a. Private Key (Symmetric) Cryptography

This mechanism also called conventional cryptography. The sender and the receiver use the same key. Sender use the key for encryption and receiver use it for decryption. The key must be known to both sender and receiver. For this reason the key must be distributed over a secure channel. The encryption key can be calculated from the decryption key and vice versa [4]. Some private key algorithms are DES, IDEA, RC4, and RC5. These algorithms are described in detail in [4, 5, 7, 8].

As shown in Figure 4, the sender encrypts the plaintext with the key and sends the ciphertext to the receiver. The receiver decrypts the ciphertext by using the same key and obtains the plaintext. The key is distributed in a secure channel.

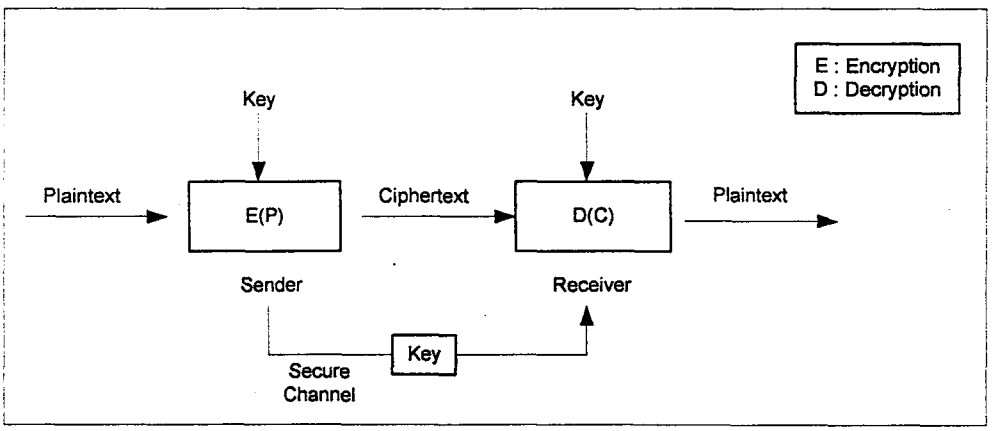

Figure 4. Private key cryptography.

b. One Way Hash Functions

Hash functions have many names: message digest, fingerprint, compression function, cryptographic checksum. A hash function is a function, mathematical or otherwise, that takes a variable-length input string and converts it to a fixed-length output string. The hash function is public; there is no secrecy to the process. The security of a one way hash function is its one-wayness. Given a hash value, it is computationally unfeasible to find an input string that hashes to that value [4]. We show the hash function in Figure 5. Some hash functions are e.g. MD5 [4, 5, 9, 10] and SHA-1 [4, 5, 10, 11].

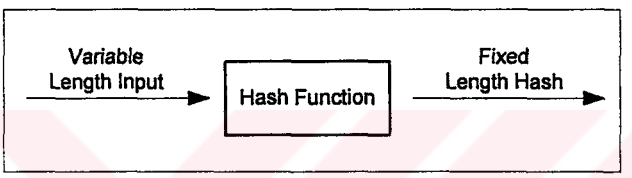

Figure 5. Hash function.

3. Authentication

Authentication allows the receiver to verify the communicating entity is the one who it claims to be. We explain authentication in two parts as user authentication and message authentication.

a. User Authentication

User authentication can be classified as one-way authentication and two-way authentication. One-way authentication is for clients. A client has to authenticate himself for logging to the system. Two-way authentication is used in mutual communication to authenticate the communicating parts. User authentication is based on three methods [12]:

(1) What You Know

To verify the identity, information like *user id* and *password* which are known by the user can be used. User id and password information which entered

by the user are compared by the system. If they are matched, the user login the system. If not, he can not login to the system. This technique proves that the password is entered correctly. But someone else can get the password. So this technique is not secure. The size of the password must be minimum 8 characters. To select password with care to reduce risk of exhaustive search is also important. By using password the system can verify the user's identity, but the user can not verify the system.

Another problem with traditional passwords is caused by eavesdropping their transfer over an insecure network. One possible solution is to use one-shot (one-time) passwords. These are passwords used only once and future values cannot be predicted from older values. Either generates a printed list, and keeps matching list on system to be accessed. Or use an algorithm based on a one-way function f (e.g. MD5 [9]) to generate previous values in series (e.g. S/KEY [13]). Passwords are generally good only for infrequent access. But they are not practical.

(2) What You Have

Because of the problems which are mentioned above, to verify the identity magnetic cards or smart cards can be used in addition to the passwords. Magnetic cards possess item with required code value encoded in it (e.g. access control cards). Smart cards may interact with the system and may require information from the user.

(3) What You Are

Here verify identity based on your physical characteristics or involuntary response patterns and also known as biometrics. Biometric keys are more secure then the knowledge (what you know) and physical keys (what you have). The main advantages of biometric keys are :

- They are unique
- It is difficult to copy

- They are always together with the user

Some biometric keys are:

- Signature (usually dynamic)
- Fingerprint
- Hand geometry
- Face or body profile
- Speech
- Retina pattern

b. Message Authentication

Message authentication allows the receiver to verify the sender's identity and ensure the integrity of the message. To achieve this message authentication code (MAC), hash function or hash-based message authentication code (HMAC) can be used.

(1) Message Authentication Code (MAC)

A public function of the message and a secret key that produces a fixed-length value that serves as the authenticator [5]. The message is encrypted with a secret key and a small fixed size block of data size is generated. This data is known as a cryptographic checksum or MAC that is appended to the message. This technique assumes that the communicating parties, suppose A and B, share a secret key. When A has a message to send B, it calculates the MAC as a function of the message and the key : $MAC=C_K(M)$. Here M is the input message, C is the MAC function, K is the secret key and MAC is the message authentication code. Message and MAC are transmitted to B. The recipient B performs the same calculation on the receiving message by using the same secret key and obtains MAC. This calculated MAC is compared with the coming MAC. If we assume that the secret key is only known by A and B, and the calculated MAC matches with coming MAC;

- The receiver is ensured that the message has not been changed. Because

an attacker can alter the message but he can not generate the proper MAC for the message. Therefore, the coming MAC and calculated MAC do not match.

- The receiver is ensured from the identity of the sender. Because no one who does not know the secret key could not prepare a message with a proper MAC.

A MAC function is similar to encryption. One difference is that the MAC algorithm need not be reversible, as it must for decryption. If we also want to provide secrecy, we can encrypt the message before or after applying the MAC algorithm.

(2) Hash Function

A different type of the message authentication code is the one-way hash function. A hash function is applied to a message and generates a fixed-size output. The message size can be different. Unlike a MAC, a hash function does not use a key. Comparison of MAC and Hash functions are shown in Figure 6. A hash function is also known as a message digest or hash value. The hash code is a function of all the bits of the message and provides an error-detection capability : a change to any bit or bits in the message results in a change to the hash code [5]. To authenticate the sender, instead of encrypting the entire message, encrypting the hash code with private or public key is adequate.

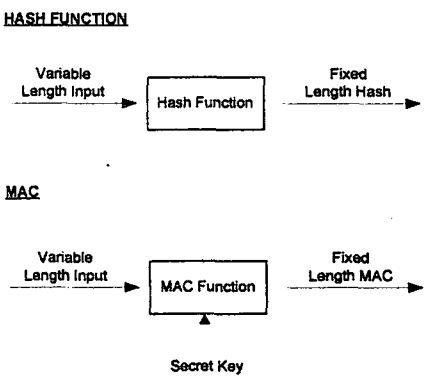

Figure 6. Comparison of Hash and MAC.

4. Hash-Based Message Authentication Code (HMAC)

HMAC [14, 15, 16] uses a cryptographic one-way hash function, such as MD5 [9] or SHA-1 [11]. The sender and the receiver share a secret key for calculation and verification of the message authentication values. The main advantages of this construction are [14]:

- Available hash functions can be used without modifications.
- The original performance of the hash function is preserved.
- Keys can be used easily.
- Authentication mechanism is realiable.
- In the case that faster or more secure hash functions are found or required, the underlying hash function can be replaced with another one easily.

a. Parameters

HMAC uses the following parameters [16]:

B → Block size (in bytes) of the input to the approved hash function.

H → An Approved hash function.

IV → Initial value input to hash function

ipad → Inner pad; the byte x'36' repeated *B* times.

K → Secret key shared between the originator and the intended receiver(s).

K_0 → The key *K* after any necessary pre-processing to form a *B* byte key.

L → Block size (in bytes) of the output of the approved hash function.

opad → Outer pad; the byte x'5c' repeated *B* times.

t → The number of bytes of MAC.

text → The data on which the HMAC is calculated; *text* does **not** include the padded key. The length of *text* is *n* bits, where $0 \leq n < 2^B - 8B$.

x'*N*' → Hexadecimal notation, where each symbol in the string '*N*' represents 4 binary bits.

|| → Concatenation

⊕ → Exclusive-Or operation.

b. Keys [16]

The size of the key, K, shall be equal to or greater than $L/2$, where L is the size of the hash function output. Note that keys greater than L bytes do not significantly increase the function strength. Applications that use keys longer than B-bytes shall first hash the key using H and then use the resultant L-byte string as the HMAC key, K. Keys shall be chosen at random using an Approved key generation method and shall be changed periodically. Note that the keys should be protected in a manner that is consistent with the value of the data that is to be protected (i.e., the *text* that is authenticated using the HMAC function).

c. Truncated Output [16]

A well-known practice with MACs is to truncate their output (i.e., the length of the MAC used is less than the length of the output of the MAC function L). Applications of this standard may truncate the output of HMAC. When a truncated HMAC is used, the t leftmost bytes of the HMAC computation shall be used as the MAC. The output length, t, shall be no less than four bytes (i.e., $4 \le t \le L$). However, t shall be at least $\frac{L}{2}$ bytes (i.e., $\frac{L}{2} \le t \le L$) unless an application or protocol makes numerous trials impractical. For example, a low bandwidth channel might prevent numerous trials on a 4 byte MAC, or a protocol might allow only a small number of invalid MAC attempts.

d. Implementation Note [14, 16]

To facilitate the implementation of HMAC, the intermediate results of the compression function on the B-byte blocks (K ⊕ opad) and (K ⊕ ipad) can be precomputed only once at the time of generation of the key K, or before its first use. These intermediate results can be stored and then used to initialize H each time that a message needs to be authenticated using the same key. For each authenticated message using the key K, this method saves the application of the hash function of H on two B-byte blocks (i.e., on $(K \oplus ipad)$ and $(K \oplus opad)$). This saving may be significant when authenticating short streams of data. These

stored intermediate values shall be treated and protected in the same manner as secret keys. Choosing to implement HMAC in this manner has no effect on interoperability.

e. Security of HMAC [15]

The security of HMAC is related to the cryptographic strength of the underlying hash function. The probability of successful attack on HMAC is equivalent to one of the following attacks on the embedded hash function:

- The attacker is able to compute an output of the compression function even with an IV that is random, secret, and unknown to the attacker.
- The attacker finds collisions in the hash function even when the IV is random and secret.

In the first attack, either a brute-force attack on the key, which is a level of effort on the order of 2^n, or a birthday attack are required.

In the second attack, the attacker is looking for two messages M and M' that produce the same hash: H(M)=H(M'). This is the birthday-attack and requires a level of effort of $2^{n/2}$ for a hash length of n. On this basis, the security of MD5 is called into question, because a level of effort of 2^{64} looks feasible with today's technology. Does this mean that a 128-bit hash function such as MD5 is unsuitable for HMAC? The answer is no, because of the following argument. To attack MD5, the attacker can choose any set of messages and work on these off line on a dedicated computing facility to find a collision. Because the attacker knows the hash algorithm and the default IV, the attacker can generate the hash code for each of the messages that the attacker generates. However, when attacking HMAC, the attacker can not generate message/code pairs offline because the attacker does not know K. Therefore, the attacker must observe a sequence of messages generated by HMAC under the same key and perform the attack on these known messages. For a hash code length of 128 bits, this requires 2^{64} observed blocks (2^{73} bits) generated using the same key. On a 1-Gbps link, one

would need to observe a continuous stream of messages with no change in key for about 250,000 years in order to succeed. Thus, if speed is concern, it is fully acceptable to use MD5 rather than SHA-1 as the embedded hash function for HMAC.

f. HMAC Specification [16]

To compute a MAC over the data '*text*' using the HMAC function, the following operation is performed:

$$MAC(text)_t = HMAC(K, text)_t = H((K_0 \oplus opad) || H((K_0 \oplus ipad) || text))_t$$

Table 1 illustrates the step by step process in the HMAC algorithm, which is depicted in Figure 7. For illustration purposes, HMAC examples are shown in appendix-1.

Table-1 The HMAC algorithm [16].

STEPS	STEP-BY-STEP DESCRIPTION
Step 1	If the length of $K = B$: set $K_0 = K$. Go to step 4.
Step 2	If the length of $K > B$: hash K to obtain an L byte string, then append $(B-L)$ zeros to create a B-byte string K_0 (i.e., $K_0 = H(K) \| 00...00$). Go to step 4.
Step 3	If the length of $K < B$: append zeros to the end of K to create a B-byte string K_0 (e.g., if K is 20 bytes in length and $B = 64$, then K will be appended with 44 zero bytes 0x00).
Step 4	Exclusive-Or K_0 with *ipad* to produce a B-byte string: $K_0 \oplus ipad$.
Step 5	Append the stream of data '*text*' to the string resulting from step 4: $(K_0 \oplus ipad) \| text$.
Step 6	Apply H to the stream generated in step 5: $H((K_0 \oplus ipad) \| text)$.
Step 7	Exclusive-Or K_0 with *opad*: $K_0 \oplus opad$.
Step 8	Append the result from step 6 to step 7: $(K_0 \oplus opad) \| H((K_0 \oplus ipad) \| text)$.
Step 9	Apply H to the result from step 8: $H((K_0 \oplus opad) \| H((K_0 \oplus ipad) \| text))$.
Step 10	Select the leftmost t bytes of the result of step 9 as the MAC.

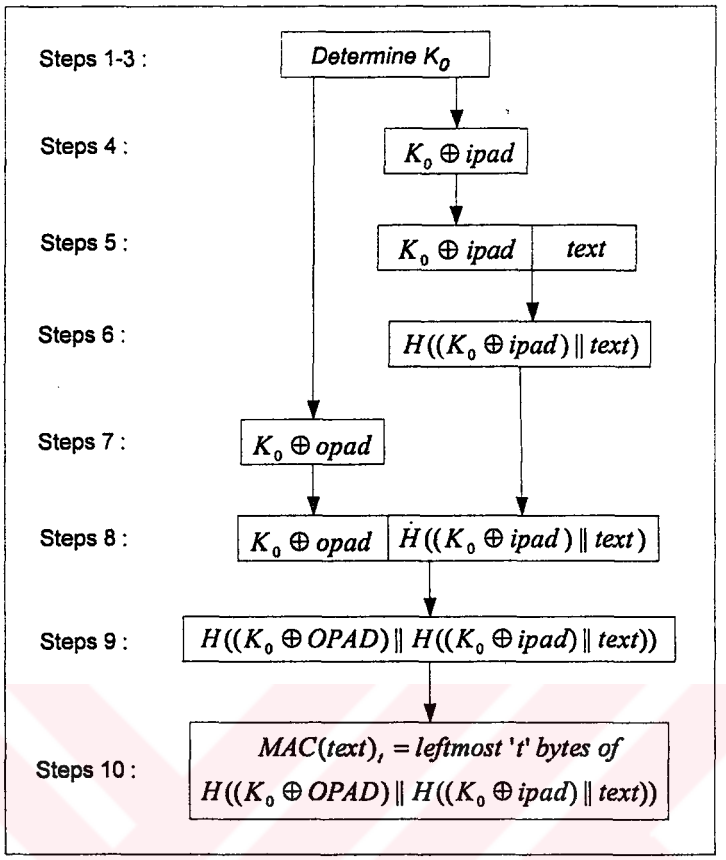

Figure 7. Illustration of the HMAC Construction [16].

B. SECURITY REQUIREMENTS FOR SENSOR NETWORKS

Security issues are key issues in wireless sensor networks where sheer number of tiny sensor nodes are scattered in a sensor field randomly. Security requirements of a secure network are outlined below [17, 18] :

Availability assures that the services of the sensor networks are available despite the denial of service attacks.

Confidentiality assures that the communication between sensor nodes should never be disclosed by another party.

Authentication assures that the receiver sensor needs verifying the sender's identity.

Integrity assures that the data content is not modified accidentally or by an adversary sensor on purpose.

Non-Repudiation assures that neither the sender nor the receiver of a message cannot be able to deny having transmitted the message.

Data Freshness implies that the data is recent, and it ensures that no adversary replayed old messages [17].

C. SECURITY VULNERABILITIES IN SENSOR NETWORKS

In this section possible attacks to sensor networks are analyzed.

1. Radio Jamming

Sensors can be prevented from communicating truthfully by jamming the radio frequencies they use. Spread spectrum, frequency hopping and region mapping methods can be used for defense. In region mapping method, nodes along the edge of a jammed region report the attack to their neighbors. Neighboring nodes collaborate to map the jamming reports, then reroute traffic around the jammed region [19, 20].

2. Battery Exhaustion

Sensor nodes can be scattered in a hostile environment, therefore their battery can not be changed. Thus, when the battery is exhausted, sensor nodes will be ineffective. One of the methods; rate limiting or giving priority to primary functions can be used for defending. In rate limiting, authors of [19], limit the MAC admission control, so that the network can ignore excessive requests without sending expensive radio transmissions. In the second method [20], they

offer to use battery management against battery exhaustion. They give priorities to functions and of course give the highest priority to the main functions. Therefore, the highest priority use of all battery.

3. Compromised Nodes

Compromised nodes may be able to reconfigure the routing protocol or any part of it. Non-repudiation is useful for detection and isolation of compromised nodes. When a node *A* receives an erroneous message from a node *B*, non-repudiation allows *A* to accuse *B* using this message and to convince other nodes that *B* is compromised [18, 20].

4. Impersonation

A hostile node controlled by the enemy may be able to join the ad hoc network undetectably and cause permanent damage to other nodes or services. We can prevent this attack with authentication.

5. Spam

Anti-nodes deployed inside a wireless sensor network can frequently generate dummy data packets that make the nodes relaying them deplete their energy and also increase the traffic load in the network. We call these types of attacks as spam attacks. A defense mechanism is explained in section IV.

D. SECURITY SOLUTIONS IN SENSOR AND AD HOC NETWORKS

Related works deal with security issues in wireless sensor networks are surveyed.

SPINS (Security protocols for sensor networks) is one of the security schemes proposed for sensor networks [17]. In SPINS two secure building blocks are used: SNEP (Secure Network Encryption Protocol) and μTESLA (the micro version of the Timed, Efficient, Streaming, Loss-tolerant Authentication

Protocol). SNEP provides data confidentiality, two party data authentication and data freshness. μTESLA provides authenticated streaming broadcast. μTESLA requires that the base station and the nodes are loosely time synchronized. In μTESLA, to send an authenticated packet, the base station simply computes a MAC on the packet with a key that is secret at that point of time. When the node gets the packet, it can verify that corresponding MAC key was not yet disclosed by the base station (based on its loosely synchronized clock and because it knows the time schedule at which keys are disclosed). Since the receiving node is assured that the MAC key is only known by the base station, it is assured that no adversary could have altered the packet in transit. So the node stores the packet in a buffer. At the time of key disclosure, the base station broadcasts the verification key to all receivers. When the node receives the disclosed key, it can easily verify the authenticity of the key. If the key is authentic, the node can now use it to authenticate the packet stored in its buffer. In [17], RC5 block cipher is also used because of the small code size and high efficiency to encrypt the data.

A Denial of Service (DoS) attack is any event that diminishes a network capacity to perform its expected function. In a typical sensor network each layer's vulnerabilities and defense mechanisms to DoS attacks are discussed in [19]. Protocol or design level vulnerabilities are considered primarily. They have examined the vulnerability of two sensor network to DoS attacks. After analyzing the vulnerabilities, they believe that developers should consider DoS susceptibility when designing new protocols.

Security, network bandwidth and power consumption in sensor networks are discussed in [21] where two applications have been implemented: target tracking and light sensing.

Routing security in wireless sensor networks is introduced in [22]. They propose security goals for routing in sensor networks, discuss how attacks against adhoc and peer-to-peer networks can be adapted into powerful attacks against

sensor networks, introduce two classes of novel attacks against sensor networks-sink holes and HELLO floods, and analyze the security of all the major proposed sensor network routing protocols. But they leave it as an open problem to design a sensor network routing protocol that satisfies their proposed security goals.

Various security schemes are introduced for adhoc networks in [18, 20, 23, 24]. The security threats an adhoc network faces and the security objectives that need to be achieved are presented in [18]. They focus on how to secure routing and how to establish a secure key management in an adhoc networking environment. To build a highly available and highly secure key management service, they propose to use threshold cryptography to distribute trust among a set of servers. Furthermore, they implement a prototype of the key management service to show its feasibility.

The principal security issues of adhoc networks are investigated and the resurrecting duckling security policy model which describes secure transient association of a device with multiple serialized owners is presented in [20].

The security of routing is also discussed in [23]. They present security problems encountered when the traditional networking approaches are applied in adhoc networking. They also give an overview of the contemporary solutions for the adhoc networking and discuss the applicability of their security architecture.

The threats and possible solutions for the basic mechanisms and for the security mechanism in mobile adhoc networks are surveyed in [24].

III. DETECT AND DEFEND SPAM (DADS) PROTOCOL

A. SENSOR WARS AND SPAM ATTACKS

The aim of sensor war is to neutralize a sensor network with another sensor network. To manage this when a sensor network is determined, another network which is similar to a sensor network is deployed close to the targeted sensor network. The second network contains anti-nodes. The number of anti-nodes is much less than the sensor nodes in the network, because they do not need to cover the entire sensor field, and of course these anti-nodes do not need to sense. We just use the communication part of the sensor nodes. To make a sensor network which is deployed in an unreachable area ineffective, using anti-nodes will be intelligent due to the natures of sensor networks discussed in section-I.A.1.

Anti-nodes are scattered randomly inside the targeted wireless sensor network. Spam attacks are set by anti-nodes by generating frequent dummy and unsolicited messages and broadcast them to the neighboring nodes in the targeted sensor network. Hence they increase the data traffic conveyed in the network. All the data generated by the nodes are forwarded to the sink. Number of the nodes close to the sink is limited and they relay much more messages than the other nodes. Therefore, sensors close to the sink are expected to fail earlier than the other sensor nodes in the network due to energy depletion. This causes the sink to be disconnected from the sensor network. If the attack continues, other sensor nodes exhaust their batteries too.

Anti-nodes may be fixed or mobile. They can use a fixed local identification or change their identifications as frequent as they need. It is relatively easier to develop a procedure to refuse the messages produced by a fixed anti-node then a mobile anti-node. For example, after detecting a spam attack, prohibiting the sensor with that id from sending any other messages to the network can be a solution. Therefore, we focus on the counter-measures against

mobile anti-nodes that change their local identifications continuously which is a more challenging case.

B. COUNTER-MEASURES AGAINST SPAM ATTACKS

In this section, we explain how to defend against spam attacks. Our scheme mainly consists of the following processes: detecting spam messages, surrounding the anti-node by putting it in a quarantine region, authenticating the messages in the quarantine region and canceling the quarantine region after neutralizing the anti-node. Refer to the appendix-2 for schematic diagram of these processes.

1. Detecting Unsolicited Messages and Anti-Nodes

A sink can detect unsolicited messages generated by anti-nodes in two ways. The first method is to filter incoming messages according to their contents and detect the nodes which send faulty messages frequently. Faulty messages can be detected by checking the contradiction between the messages sent by neighboring nodes. The second method is to check the frequency of messages sent by the sensors in the same region. If a node sends messages more frequently than its neighbors, it is reasonable to suspect that the node is an anti-node.

Since sensor nodes have some shortages due to cost and size, they cannot detect and filter messages coming from anti-nodes. In our scheme the sink detects the messages sent by an anti-node. Sink can detect faulty messages by using the first method explained above. However the first method is not practical because anti-nodes can listen to the sensor nodes and repeat their messages after changing the required fields such as id, time, etc. Therefore, applying the second method is a better approach because it fits the characteristics of spam attacks naturally. In this method, detection mechanism is based on checking the frequencies of the messages generated by the sensor nodes. If a sensor node in a region generates messages more frequently than the other nodes in the same region, i.e., δ times

more messages where δ is a system parameter, it may indicate that it is an anti-node.

2. Determining the Borders of a Quarantine Region

In our scheme, authentication is not required in a typical message. The fields of a typical message are shown in Figure 8. *Source id* is the local identification of the node that generates the sensed data, i.e., source node. *Source location* is the location of the source node according to a coordinate system, e.g., polar coordinates, grid coordinates, etc. Location awareness of sensor nodes is generally a requirement for especially tactical sensor network applications. For example, a target detection data is almost meaningless without associating it with a location. There are a number of GPS based, beacon based and beaconless location estimation schemes [25, 26] applicable to the tactical sensor networks. Therefore, the assumption that sensor nodes know their location is practical. *Last hop node* is the node that relays the message. *Last hop location* is the location of the *last hop node*. Every node that relays a message replaces this field with its location. The *last hop location* is the same as the *source location* when the message is transmitted by the source node. *Sensed data* is the payload in the message. Of course, some extra fields may exist in actual message which are not relevant to our protocol. For applying our protocol, the typical message must contain all of these fields.

source id	source location	last hop node	last hop location
sensed data			

Figure 8. The format of a typical message.

The sensor node first compares its location with the *last hop location* in an incoming message, and does not repeat the message unless the *last hop location* is closer than the *maximum relay distance* d_{max} which is a system

parameter. The *maximum relay distance* d_{max} is longer than the *transmission range* r of sensor nodes, and can be found by

$$d_{max} = \theta \times r \quad (1)$$

where θ is a multiplication factor.

The center of a quarantine region is obtained from the *source location* field of a spam message. The size of a quarantine region is a function of *maximum relay distance* d_{max}. Quarantine region is a square shaped region that has an anti-node at its center as shown in Figure 9. The distance d_q between the borders of a quarantine region and the anti-node is given by

$$d_q = \beta \times d_{max} \quad (2)$$

where β is a system parameter greater than 1.

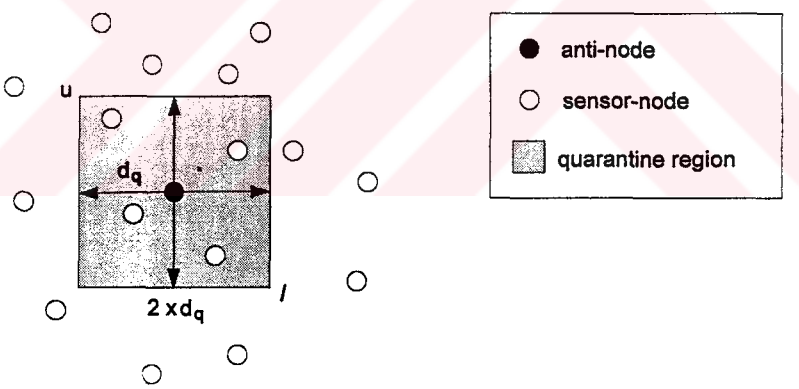

Figure 9. Boundaries of a quarantine region.

By using d_q, x and y coordinates of the anti-node the coordinates of upper left u and lower right l corners of the quarantine region can be found as shown in Figure 9. Then these coordinates u and l are broadcasted to the sensor network by the sink. Hence sensor nodes can find out whether a location is within the quarantine region or not. In the performance evaluation section, we evaluate the

performance of our scheme also by factoring d_q value to examine the sensitivity of our scheme against this parameter.

3. Authentication in a Quarantine Region

Figure 10 shows a sensor field where a quarantine region is indicated by the gray area. The nodes 3, 4, 7, 8 and an anti-node are in the quarantine region therefore they have to send authenticated messages. When a sensor has a message to send, it first checks if it is in one of the quarantine regions. If so, it sends the message authenticated. Nodes do not relay any unauthenticated messages whose *last hop location* is in one of the quarantine regions. For example, nodes 3 and 4 do not relay messages coming from the anti-node if they are not authenticated. Similarly, node 7 does not relay unauthenticated messages coming from nodes 3 and 4, and node 11 does not relay unauthenticated messages coming from nodes 7 and 8. Nodes outside the quarantine regions do not need authentication to transmit a message even if the message was an originally authenticated message coming from a quarantine region. For example, node 11 receives authenticated messages from node 8; it repeats them unauthenticated because node 8 is in the quarantine region but node 11 is not. If node 11 has a message to send for node 7, it sends the message unauthenticated. Because node 11 is not in the quarantine region, however node 7 is in the quarantine region and it relays the message to node 3 and node 4 authenticated. Also note that a particular node, n, relays only the messages coming from nodes whose locations are within a d_{max} radius with respect to the location of n.

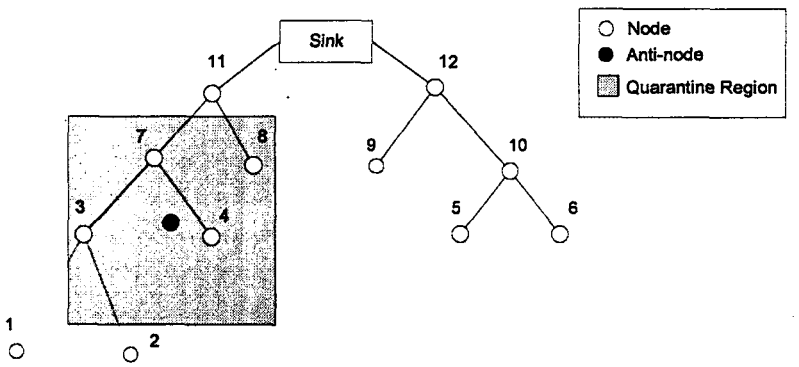

Figure 10. A simple sensor network and a quarantine region.

Sensor nodes use the proposed *detect and defend spam (DADS)* protocol for authentication. DADS must be simple enough to fit the stringent constraints of tiny sensors. Therefore it does not use asymmetric and symmetric cryptography, but only cryptographic hash functions. Therefore, we use the standard HMAC (hash-based message authentication code) mechanism [14, 15] for message authentication.

HMAC uses a cryptographic one-way hash function, such as MD5 [9]. The sender and the receiver share a secret key, *K*. The message authentication code over the message, *M*, is calculated as,

$$\text{HMAC} = H(K \oplus \text{opad} \,||\, H(K \oplus \text{ipad} \,||\, M) \qquad (3)$$

where $\oplus$ is the bitwise *"exclusive OR"* operation, *H* is the underlying one-way hash function, *ipad* and *opad* are two constants defined in [14] and [15].

It is necessary to mention about the power consumption of HMAC algorithm. For example, the Berkeley motes [27] consume 1 μJ for transmitting and 0.5 μJ for receiving a single bit, while the CPU can execute 208 cycles (roughly 100 instructions) with 0.8 μJ [28]. We have written the HMAC algorithm in C and assembled it to the assembler code by AVR Studio [29]. We observe that our HMAC algorithm consumes approximately 4,5 mJ.

In DADS, an authenticated message contains the fields shown in Figure 11. *Source id*, *source location*, *last hop node id*, *last hop location* and *sensed data* fields are the same as the fields in a typical message given in Figure 8. *Sequence number* and *authentication code* fields are added to the message structure in support of authentication. *Sequence number* is the number of outgoing messages and *authentication code* is the HMAC value.

source id	source location	last hop node id
last hop location	Sequence number	authentication code
sensed data		

Figure 11. An authenticated message.

It is necessary to make some assumptions about the fields in the message. Table 2 shows these fields and their size. It is assumed that the *source id* and *last hop node id* are 16 bits. It means that sensor network can contain 65535 sensor nodes. *Source location* and *last hop location* are 24 bits. We assume that degree values for latitude and longitude will not change in the sensor field. So we use 2 bits for minute and 10 bits for second. Thus we can get latitude with 12 bits, and we need 12 bits for longitude too. 12 square miles can be covered with these 24 bits. *Sequence number* is 18 bits, 4 times bigger than the number of sensor nodes in the network. 64 bits are adequate for *authentication code*. When we assume that a message is approximately 30 bytes (240 bits) [17], we have 78 bits (roughly 10 bytes) for sensed data.

Table-2 Message fields size.

Field	**Size/bits**
source id	16
source location	24
last hop node id	16
last hop location	24
sequence number	18
authentication code	64
sensed data	78

Sensors are equipped with the same secret key K before deployment. When a sender has a message to send, it first generates the *authentication code* using the HMAC algorithm and the key, K. The message over which HMAC is to be calculated contains the *source id*, *source location*, *last hop node id*, *last hop*

location, *sequence number* and *sensed data* fields. The *sequence number* is incremented for every outgoing message. After the composition, the authenticated message is transmitted. Any node that should relay this message generates the *authentication code* by using the same algorithm, message and key. If the value calculated at the end of this process is not equal to the value in the *authentication code* field of the incoming message, the message is discarded. Otherwise the message is accepted. This mechanism is depicted in Figure 12.

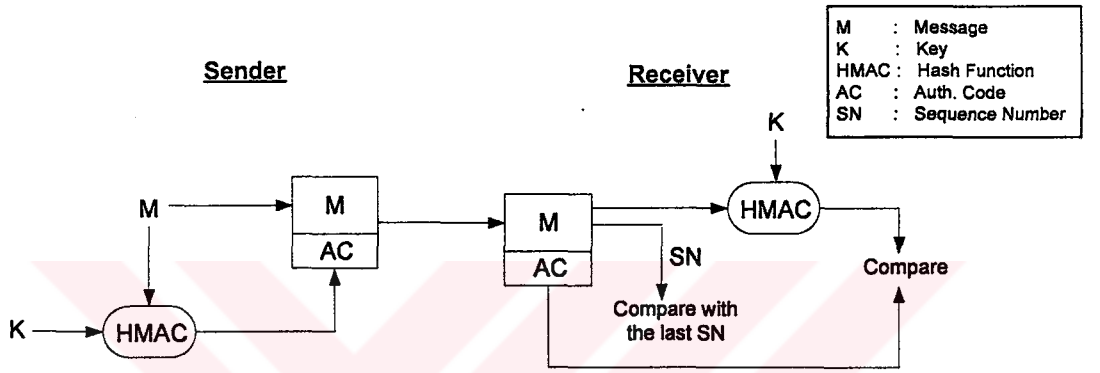

Figure 12. Message authentication in DADS.

To facilitate the implementation of HMAC in the sensor nodes, (K ⊕ opad) and (K ⊕ ipad) can be precomputed as offered in [14] and [15]. This implementation is more efficient especially when the message is short. As discussed in [17], sensor nodes use very small messages (approximately 30 bytes). Thus, this implementation is suitable for DADS.

DADS thwarts possible replay attacks of anti-nodes using the *sequence numbers*. Every sensor node has a *sequence number* to be used for sequence numbers in outgoing messages. It begins with zero and is incremented by one for each message. Thus, the *sequence number* of a message created or relayed by a sensor node should be greater than that of every message sent or relayed by the same node before. Each sensor node keeps the last *sequence number* obtained from each of its neighboring nodes. The freshness of each received message is checked by comparing the *sequence number* in the received message with the last

sequence number of the *last hop node id* of that message that is kept locally. If the received message has a higher *sequence number* and *authentication code* is verified, then it is concluded that the message is not a replay and authentic. Such a message is accepted for relaying and the locally kept last *sequence number* is updated accordingly. Relaying nodes do not accept a message with a *sequence number* which is equal or less than the preceding ones. In such a case, relaying node asks the *authentication code* for the same message but with the expected *sequence number*. If the last node cannot regenerate this *authentication code*, then the message is discarded.

We assume that the sequence number is long enough that it never repeats within the lifetime of the node. This is an acceptable assumption made also in other well known paper [17]. Since we do not always use authentication but a node needs authentication only when it is in a quarantine region; we have advantages in making this assumption comparing to the other technique [17].

One may argue that the *authentication codes* created by a sensor node, *s*, hidden to another node, *n*, can be exploited by anti-nodes. Since those authenticated messages are not received by *n*, the anti-nodes can record and later resend them to *n*. The node *n* accepts those replays as valid and relays them. However anti-nodes can never reach to a significant spam rate by using any of these techniques, because they need to keep pace with the other nodes to use the *authentication codes* generated by them.

One may also try to capture a node and obtain the key by a physical examination. However, DADS aims to prevent spam attacks for the sensor networks deployed in physically inaccessible regions. Moreover, the lifetime of a sensor network in a battlefield is too limited for tampering with a sensor node and obtaining the key out of it.

4. Canceling a Quarantine Region

As the declaration of the quarantine regions, cancellation of them is also the responsibility of the sink. Sink makes the decision to cancel a quarantine region based on the *periodic status reports* sent by the nodes. If the sink does not receive *a spam report* during a time period called *status report period* for a quarantine region, it cancels that quarantine region. Sensor nodes generate spam reports periodically and independent from each other. When a sensor node receives a message with no or invalid *authentication code* from a quarantine region, it sets the *spam-continues flag* for that quarantine region. At the end of *status report period*, each sensor node generates a spam report that includes all the quarantine regions that has a spam-continues flag set, and then removes all the flags and starts the next status report period. The nodes that relay spam reports read the content of the report. If they have a flag set for a quarantine region in the report, they remove the flag for that quarantine region and relay the report. Thus sensor nodes produce minimum number of spam reports. We propose this method in order to prevent the possible increase in network traffic because of spam reports. If the sink does not receive a spam report for a quarantine region during a specific *status report period*, it cancels that quarantine region by broadcasting a "cancel quarantine region" command. Note that the sink and sensor nodes do not need to synchronize their *status report periods*. The sink may receive a spam report for a quarantine region just after canceling it due to the lack of synchronization. In such a case, the sink re-declares the region as a quarantine region.

We want to attract attention to the communication errors. A legal node can take incorrect messages from another legal node, because of the communication errors. At that time, the first node will think that the received message is spam and will generate a spam report. Thus, the quarantine region may not be cancelled. This misunderstanding can be solved by error-correction methods. But this is beyond our scope.

IV. PERFORMANCE EVALUATION

In our simulations, we examine the increase in data traffic due to spam attacks and analyze the performance of DADS. In our model, 100 sensor nodes and 10 anti-nodes are deployed in a sensor field 200×200 units in size. One of the topologies used in our simulation is shown in Figure 13. In the figure, the sensor nodes are depicted with '+' and the anti-nodes are depicted with '*'. We use directed diffusion [30] as the data dissemination scheme. The paths from sensors to the sink are shown with straight lines. We assume that the sensor nodes are immobile and location aware. We use Matlab 6.0 [31] for simulation. Refer to the appendix-3 for pseudocode of the simulation program.

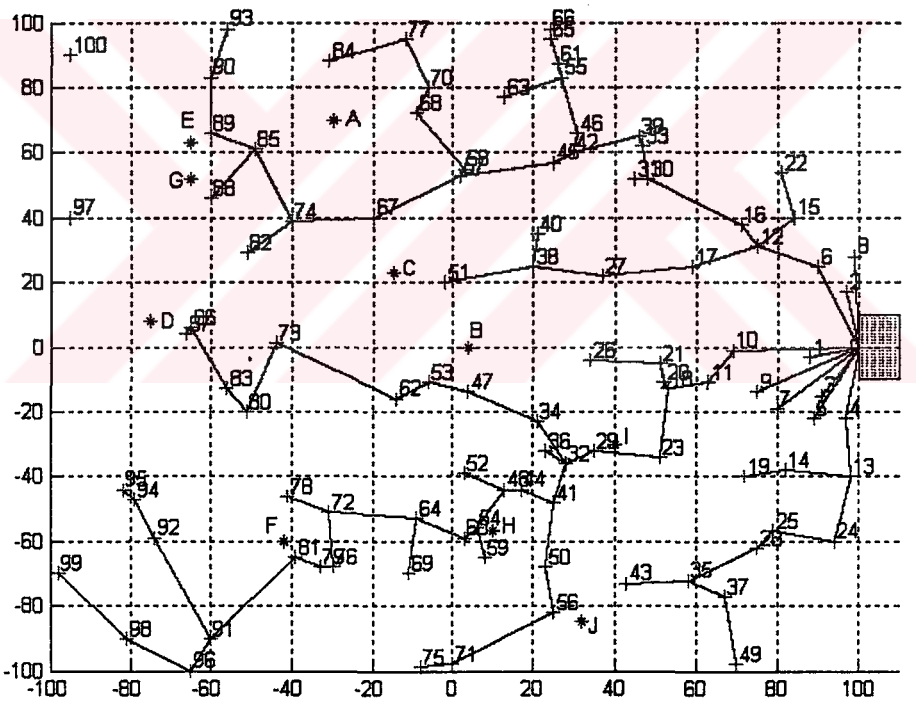

Figure 13. Sensor and anti-node deployment in a sensor field.

We first analyzed the network traffic in the sensor field. In Figure 14, number of hops required to reach the sink is shown for each sensor node. The total number of required hops of all sensor nodes in the field is 832; that is, if each node sends one message to the sink in one period, 832 hops will be created. We

observe almost linear increase in the number of hops as the sensor node goes further from the sink. Therefore not only the number of anti-nodes but also the location of them is important in the impact of spam attacks; the anti-nodes further from the sink create higher faulty traffic.

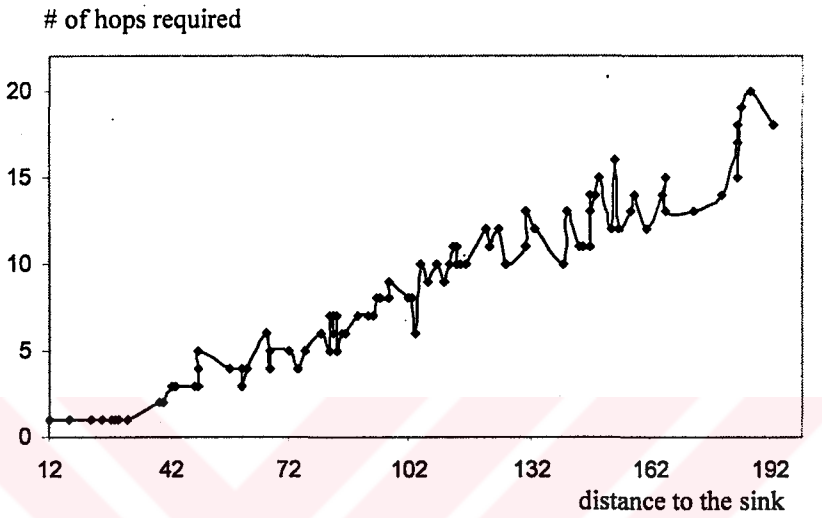

Figure 14. The number of hops required to send a message to the sink.

We also analyzed number of hops required for a spam message generated by an anti-node to reach the sink. After an anti-node broadcasts a spam message, all of the nodes to which this message is directed try to relay it to their gradients so that the spam message reaches to the sink eventually. Table 3 shows the total number of hops required to deliver a message generated by an anti node to the sink. For instance, when Anti-node A sends a spam message, 9 sensor nodes (67, 68, 70, 74, 77, 84, 85, 89, 90) receive this message and repeat it to the sink by creating 110 hops. In other words, when Anti-node A sends a spam message, it causes each of the 9 neighboring sensor nodes to send one message to the sink. If all anti-nodes send one message in one period, a total of 781 hops are created. When the number of messages caused by sensor nodes and anti-nodes are compared, it is easy to see that anti-nodes can increase the number of messages in the network dramatically.

Table-3 Sensors affected by anti-nodes.

Anti-node id	Sensors affected by an anti-node	Hop Count
A	67,68,70,74,77,84,85,89,90	110
B	26,34,38,47,51,53,62	51
C	51,57,67,74	36
D	73,82,83,86,87	65
E	74,85,88,89,90	63
F	64,69,72,76,78,79,81,91,92	132
G	74,82,85,88,89,90,97	75
H	32,36,41,44,48,50,52,54,56,59,60,64,69	121
I	11,18,19,20,21,23,26,29,32,34,36,41,44,48	77
J	35,43,50,56,59,71	51
	TOTAL :	781

In Figure 15 we show the change in the number of hops versus the number of sensor nodes in a sensor field where the field size, sensor node transceiver range and number of anti-nodes are all fixed. At the beginning, there are 100 sensor nodes in the sensor field which causes 781 total hops. If we increase the number of sensor nodes 5 times and deploy 500 sensor nodes in the sensor field, number of hops increases approximately 12 times as shown in Figure 15. This proves that anti-nodes can be more effective in the sensor network where the node density is high.

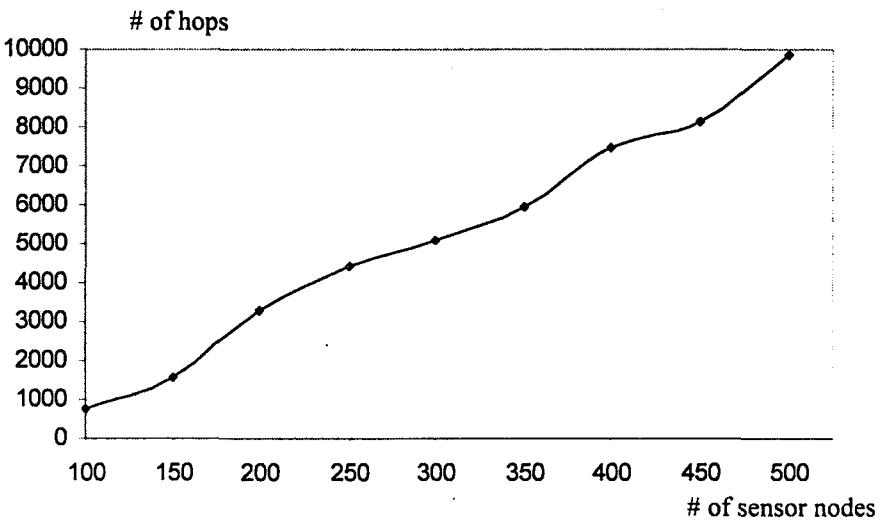

Figure 15. Impact of the sensor density.

We expect that the spam attacks become more effective when the number of anti-nodes in the sensor field increases. We have analyzed this situation by increasing the anti-node ratio in the network. Anti-node ratio is the total number of anti-nodes divided by the total number of sensor nodes in the sensor field. As shown in Figure 16, the increase in the anti-node ratio causes a lot of hops and DADS scheme diminishes the damage of anti-nodes, especially when the anti-node ratio is high.

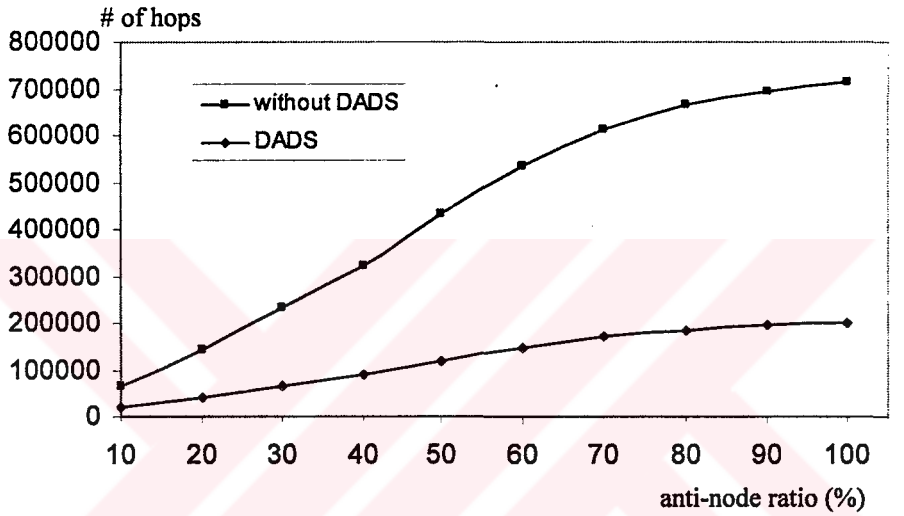

Figure 16. Impact of increase in the anti-node ratio with and without DADS.

In Figure 17, we depict how DADS prevents spam traffic effectively. We simulate the sensor network with 100 sensor nodes and 1 to 10 anti-nodes. We assume that every node sends 100 messages. As shown in Figure 17, in the average the proposed DADS scheme eliminates 72 % of the traffic caused by anti-nodes. We calculate the percentage as,

$$\left(\sum_{i=1}^{n}\left((wD_i - D_i)/wD_i\right)*100\right)/n \qquad (4)$$

where n is number of anti-nodes, wD is without DADS value and D is the value which DADS is used..

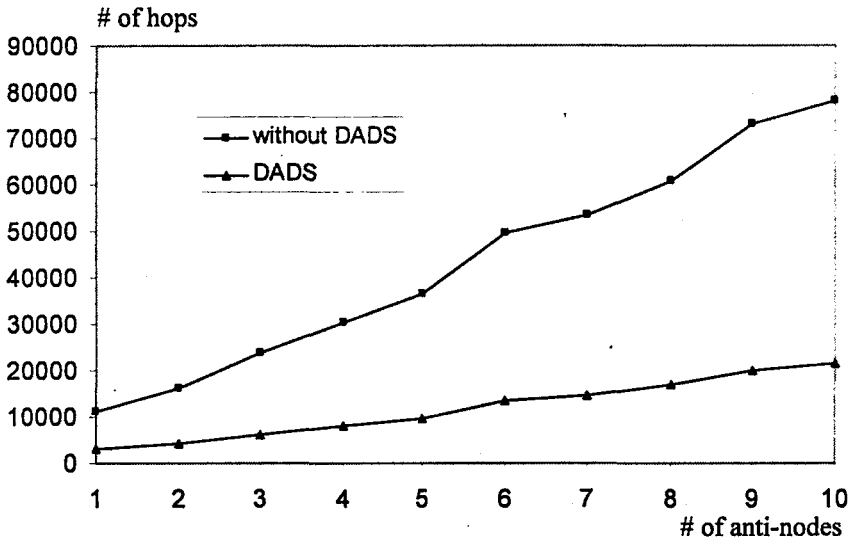

Figure 17. Effect of DADS in a sensor field with 100 sensor nodes and some anti-nodes.

Figure 18 depicts the number of authenticated hops versus the number of anti-nodes for different values of d_q, distance between the anti-node and the borderline of the quarantine region. The higher the distance d_q is, the more authenticated hops in the quarantined region, because when d_q is higher, quarantine regions are bigger.

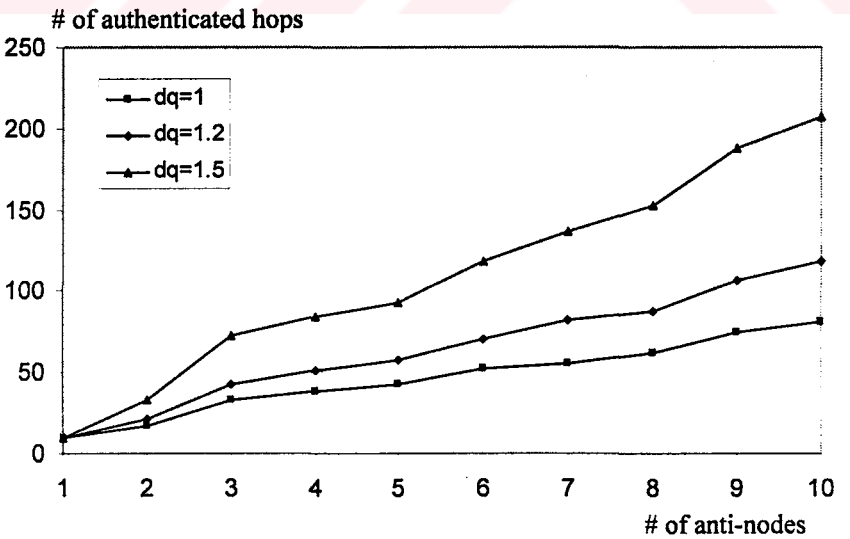

Figure 18. Sensitivity against to the changes in d_q.

In Figure 19, we depict the effect of spam frequency in DADS. Network performances with and without DADS are very close to each other when the number of spam messages per anti-node is small. It takes some time to detect that there is a spam attack. Therefore when the number of spam messages per anti-node is limited, the sink does not receive enough number of spam messages to detect a spam attack. Moreover the percentage of traffic eliminated by the DADS scheme increases as the number of messages generated by anti-nodes increases.

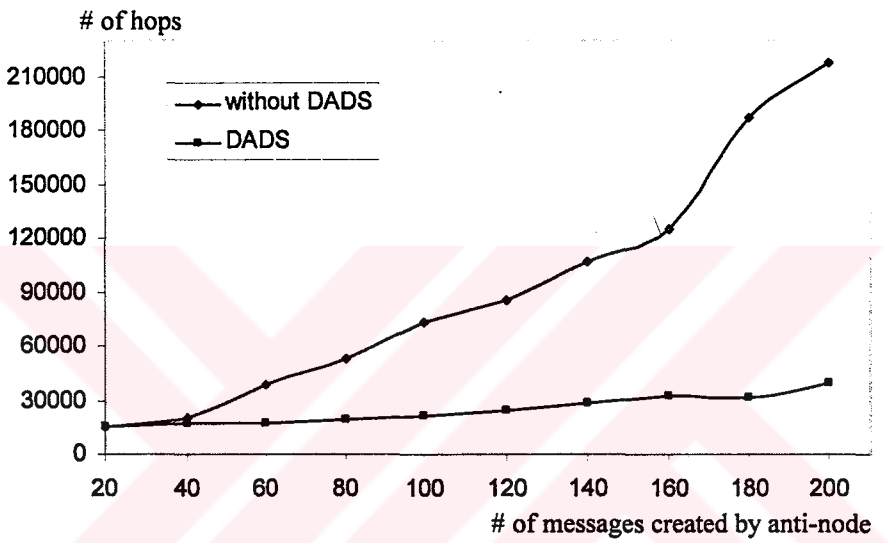

Figure 19. Impact of the spam frequency.

In Figure 20, we show the number of nodes in quarantine regions versus the number of authenticated hops in the quarantine regions. There are 79 sensor nodes in all quarantine regions. But some sensor nodes are affected by more than one anti-nodes and therefore are included in more than one quarantine regions. Thus the total number of distinct sensor nodes in all quarantine regions is 54 out of 100 total nodes in the sensor field. Actually the number of authenticated hops in a quarantine region is very small. But, sensor nodes in a quarantine region have to relay the messages authenticated which are coming from the outside of the quarantine region. Especially the nodes closer to the sink relay much more messages. Therefore, the quarantine regions closer to the sink cause more

authenticated messages than the other ones. We can see this increase in the number of authenticated hops from 203 to 470.

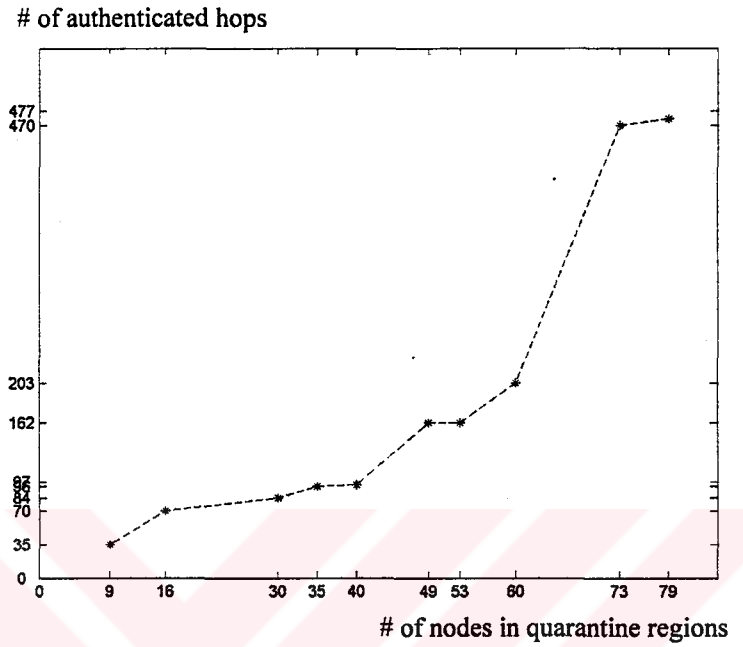

Figure 20. Impact of the number of nodes in quarantine regions.

We show the effect of the number of anti-nodes on the cost of DADS in Figure 21. When we compare spam messages with authenticated messages, we observe that sending 41% of the messages authenticated are adequate for impeding spam messages in average for any number of anti-nodes. We show the average percentage of authenticated hops with an intermittent line in the figure. These authenticated hops are the cost of using DADS protocol. If DADS is not used, 10 anti-nodes cause 781 hops. When DADS is used, spam messages are impeded. However sensor nodes in quarantine regions have to send their messages authenticated.

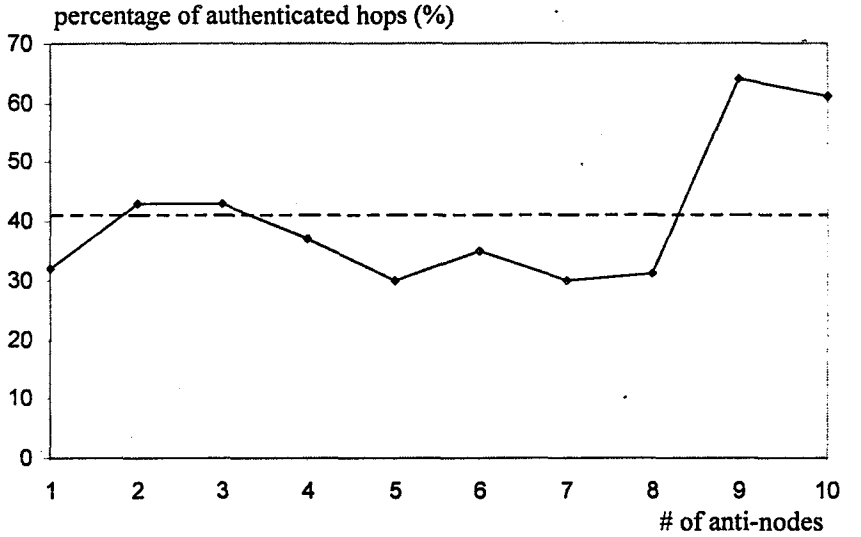

Figure 21. Number of authenticated hops in DADS.

We depict the area of the quarantined regions versus the number of anti-nodes in the sensor field in Figure 22. 10 anti-nodes create quarantine regions of 19385 square-units that correspond to 48% of the sensor field. In other words, in order to neutralize the anti-nodes, only 48% of the whole sensor field should send authenticated messages; the remaining 52% does not have to take the burden of authenticating messages.

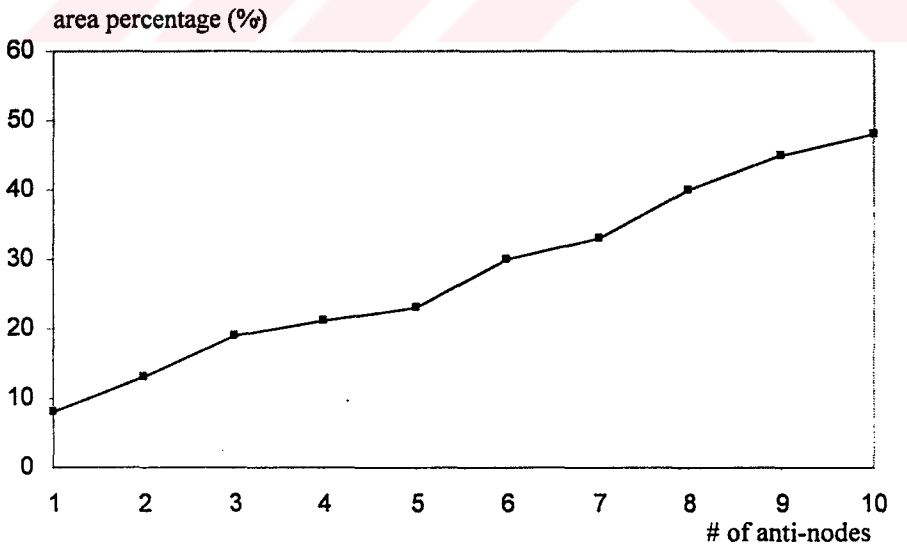

Figure 22. The percentage of the quarantined regions with respect to the number of anti-nodes.

We want to observe the effect of different anti-node deployments to the number of authenticated hops. Instead of deploying the anti-nodes with random patterns, we deployed the anti-nodes with some regular patterns. The patterns are shown in Figure 23. In all cases, the anti-nodes are depicted with '*' and the sink is depicted with '■'. Case-2, case-3 and case-5 cause the most authenticated message relay in the network. Sensor nodes in a quarantine region have to relay the messages authenticated which are coming from the outside of the quarantine region. Therefore, the quarantine regions closer to the sink cause to send more authenticated messages than the other nodes. We have observed that the affect of the anti-nodes are increased when they are deployed with a regular pattern.

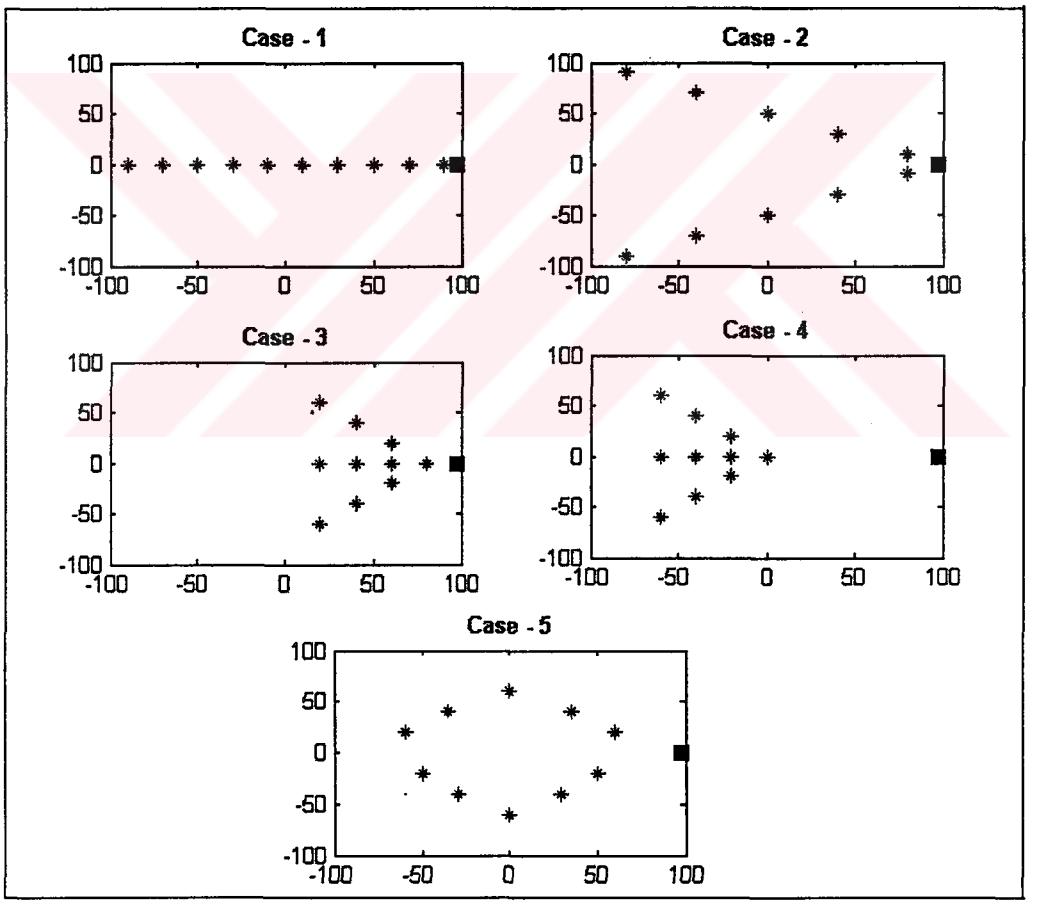

Figure 23. Different anti-node deployments.

V. CONCLUSION

A *sensor network* is a collection of sheer number of sensor nodes that collaboratively work by using multi-hop wireless communications architecture. Sensor networks have wide-range of applications especially in a battlefield because of their flexible, low cost, and self organizing features.

Security issues are of the key issues especially in wireless sensor networks often deployed beyond the enemy lines. When they are reachable, they can be collected and/or destroyed by the enemy. If they are deployed in a hazardous region or a region not accessible for a time period, the defender can deploy some anti-nodes in the tactical sensor network to create spam traffic that causes the sensor nodes to waste their energy by relaying those spam messages towards the sink. We call this attack as spam attack. If a sensor network is not protected against spam attacks properly, even a few anti-nodes lessen the sensor network lifetime. One solution for spam attack problem is to use authentication mechanism in the sensor network. In this method, normally, all messages in the network have to be sent authenticated to prevent anti-nodes from sending messages. The cost of authenticating all messages through sensor network is high. DADS (detect and defend spam) scheme lessens this cost effectively by using local quarantine regions. Authentication mechanism is applied in these local areas. Hence, authentication mechanism is not a burden to all of the sensor nodes in the network. Naturally, authentication in quarantine regions still cost, but this cost can be ignored and there is no free lunch.

In this thesis, we introduce DADS protocol as a countermeasure for spam attacks. The sink of the sensor network detects spam messages by checking the frequency of messages sent by the sensor nodes. If an anti-node is spotted in this way, then the sink declares a quarantine region, which is a square shaped region that has the anti-node at its center. The sensor nodes in the quarantine region have to send the messages in authenticated manner. Since the anti-node in the

quarantine region cannot generate authenticated messages, anti-node's messages are discarded by the relaying sensor nodes. Thus, the spam messages are isolated within the quarantine region. In this way, the other nodes in the sensor network are saved from the burden of relaying spam messages. If no spam message is broadcasted from the quarantine region for a specific period of time, the sink cancels the quarantine region. Hence the quarantined sensor nodes resume normal operation thereafter.

LIST OF REFERENCES

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, "Wireless Sensor Networks: A Survey", Computer Networks Journal (Elsevier), Vol. 38, No.4, pp. 393-422, March 2002.

[2] http://tinyos.millennium.berkeley.edu

[3] W. Diffie and M. Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory, pages 644-654, 1976.

[4] B. Schneier, "Applied Cryptography: Protocols, Algorithms and Source Code in C", John Wiley & Sons, New York, Second edition, 1996.

[5] W. Stallings, "Cryptography and Network Security", Prentice Hall, Second edition, 2000.

[6] M.E. Hellman, "An Overview of Public Key Cryptography", IEEE Communications Magazine, pp. 42-49, May 2002.

[7] M. Robshaw, "Stream Ciphers", RSA Laboratories Technical Report TR-701, July 1995. http://www.rsasecurity.com/rsalabs/index.html.

[8] R.L. Rivest, "The RC5 Encryption Algorithm", Proc. of First Workshop on Fast Software Encryption, pp. 86-89, 1995.

[9] R. L. Rivest, "RFC 1321 : The MD5 message-digest algorithm", April 1992.

[10] M. Robshaw, "MD2, MD4, MD5 SHA and Other Hash Functions", RSA Laboratories Technical Report TR-101, July 1995, http://www.rsasecurity.com/rsalabs/index.html.

[11] D. Eastlake, P. Jones, "RFC 3174 : US Secure Hash Algorithm 1 (SHA1)", September 2001.

[12] "Authentication, Hash Functions, Digital Signatures.htm", http://williamstallings.com/Extras/Security-Notes/lectures/authent.html.

[13] N. Haller, "RFC 1760: The S/KEY One-Time Password System", February 1995.

[14] H. Krawczyk, M. Bellare, R. Canetti, "RFC 2104: HMAC: Keyed-Hashing for Message Authentication", February 1997.

[15] W. Stallings, "Cryptography and Network Security", Prentice Hall, Third edition, 2003.

[16] National Institute for Standards and Technology (NIST), "The Keyed-Hash Message Authentication Code (HMAC)", FIPS PUB 198, March 2002.

[17] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J.D. Tygar, "SPINS: Security Protocols for Sensor Networks", Proc. of ACM MobiCom'01, pp. 189-199, Rome, Italy, 2001.

[18] L. Zhou and Z.J. Haas, "Securing Ad Hoc Networks", IEEE Network Magazine, vol. 13, no.6, pp. 24-30, November/December 1999.

[19] A.D. Wood, J.A. Stankovic, "Denial of Service in Sensor Networks", IEEE Computer Magazine, pp. 54-62, October 2002.

[20] F. Stajano, R.Anderson, "The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks", Proc. of the 7th International Workshop on Security Protocols, pp. 172-182, Berlin, 2000.

[21] M. Chen, W. Cui, V. Wen, A. Woo, "Security and Deployment Issues in a Sensor Network", December 2000.

[22] C. Karlof, D. Wagner, "Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures", to appear in 1st IEEE International Workshop on Sensor Network Protocols & Applications.

[23] V. Karpijoki, "Security in Ad Hoc Networks", Proc. of the Helsinki University of Technology Seminar on Network Security, 2000.

[24] J.P. Hubaux, L. Buttyan, S. Capkun, "The Quest For Security in Mobile Ad Hoc Networks", Proc. of ACM Symposium on Mobile AdHoc Networking and Computing (MobiHOC 2001), 2001.

[25] A. Nasipuri, K. Li, "A Directionality Based Location Discovery Scheme for Wireless Sensor Networks", Proc. of the 1st ACM Workshop on Wireless Sensor Networks and Applications, pp. 105-111, Atlanta, 2002.

[26] A. Savvides, H. Park, M.B. Srivastava, "The Bits and Flops of the N-hop Multilateration Primitive for Node Localization Problems", Proc. of the 1st ACM Workshop on Wireless Sensor Networks and Applications, pp. 112-121, 2002.

[27]] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System Architecture Directions for Network Sensors", ASPLOS 2000.

[28] S. Bhattacharya, H. Kim, S. Prabh, T.F. Abdelzaher, "Energy-Conserving Data Placement and Asynchronous Multicast in Wireless Sensor Networks", MobiSys 2003.

[29] http://www.atmel.com/products/AVR

[30] C. Intanagonwiwat, R. Govindan, D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks", Proc. of the 6th Annual International Conference on Mobile Computing and Networks (MobiCOM'00), August 2000.

[31] http://www.mathworks.com/products/matlab

[32] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", FIPS PUB 180-1, April 1995.

[33] Y. Bae Ko and N.H.Vaidya, "Geocasting in Mobile Ad Hoc Networks:Location-Based Multicast Algorithms", Proc. of IEEE WMCSA'99, New Orleans, pp. 101-110, February 1999.

[34] Y.C. Hu, A. Perrig, D.B. Johnson, "Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad Hoc Networks", Rice University technical report TR01-384, December 2001.

[35] J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway, "UMAC: Fast and Secure Message Authentication", In Advances in Cryptology - CRYPTO '99, Lecture Notes in Computer Science, Springer-Verlag, (1999). Full version of this paper, available at www.cs.ucdavis.edu/~rogaway/umac.

[36] G. Tsudik, "Message Authentication with One-Way Hash Functions", Proc. of INFOCOM'92, May 1992.

[37] S.Sancak and Ç.Çimen, "Telsiz Sensör Ağlarında Güvenlik", SAVTEK 2002, Savunma Teknolojileri Kongresi, Ekim 2002.

[38] A.M. Law, W.D. Kelton, "Simulation Modeling & Analysis", McGraw-Hill, Second edition, 1991.

[39] S. Coleri, M. Ergen, T.J. Koo, "Lifetime Analysis of a Sensor Network with Hybrid Automata Modelling", First ACM International Workshop on Wireless Sensor Networks & Applications, pp. 98-104, September 2002.

[40] Y. Zhang, W. Lee, "Intrusion Detection in Wireless Ad-Hoc Networks", MobiCom'2000, August 2000.

APPENDIX - 1 : HMAC EXAMPLES

These examples are provided in order to promote correct implementations of HMAC [16].

The SHA-1 hash function used in these examples is specified in [57].

1.1 SHA-1 with 64-Byte Key

Text : "Sample #1"

Key : 00010203 04050607 08090a0b 0c0d0e0f
10111213 14151617 18191a1b 1c1d1e1f
20212223 24252627 28292a2b 2c2d2e2f
30313233 34353637 38393a3b 3c3d3e3f

K_0 : 00010203 04050607 08090a0b 0c0d0e0f
10111213 14151617 18191a1b 1c1d1e1f
20212223 24252627 28292a2b 2c2d2e2f
30313233 34353637 38393a3b 3c3d3e3f

$K_0 \oplus$ ipad:
36373435 32333031 3e3f3c3d 3a3b3839
26272425 22232021 2e2f2c2d 2a2b2829
16171415 12131011 1e1f1c1d 1a1b1819
06070405 02030001 0e0f0c0d 0a0b0809

(Key $\oplus$ ipad) || text:
36373435 32333031 3e3f3c3d 3a3b3839
26272425 22232021 2e2f2c2d 2a2b2829
16171415 12131011 1e1f1c1d 1a1b1819
06070405 02030001 0e0f0c0d 0a0b0809
53616d70 6c652023 31

Hash((Key $\oplus$ ipad) || text):
bcc2c68c abbbf1c3 f5b05d8e 7e73a4d2
7b7e1b20

$K_0 \oplus$ opad:
5c5d5e5f 58595a5b 54555657 50515253
4c4d4e4f 48494a4b 44454647 40414243
7c7d7e7f 78797a7b 74757677 70717273
6c6d6e6f 68696a6b 64656667 60616263

($K_0 \oplus$ opad) || Hash((Key $\oplus$ ipad) || text):
5c5d5e5f 58595a5b 54555657 50515253

4c4d4e4f 48494a4b 44454647 40414243
7c7d7e7f 78797a7b 74757677 70717273
6c6d6e6f 68696a6b 64656667 60616263
bcc2c68c abbbf1c3 f5b05d8e 7e73a4d2
7b7e1b20

HMAC(Key, Text) = Hash((K$_0$ ⊕ opad) || Hash((Key ⊕ ipad) || text)):
4f4ca3d5 d68ba7cc 0a1208c9 c61e9c5d
a0403c0a

20-byte HMAC(Key, Text):
4f4ca3d5 d68ba7cc 0a1208c9 c61e9c5d
a0403c0a

1.2 SHA-1 with 20-Byte Key

Text : "Sample #2"

Key : 30313233 34353637 38393a3b 3c3d3e3f
40414243

K$_0$: 30313233 34353637 38393a3b 3c3d3e3f
40414243 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000

K$_0$ ⊕ ipad:
06070405 02030001 0e0f0c0d 0a0b0809
76777475 36363636 36363636 36363636
36363636 36363636 36363636 36363636
36363636 36363636 36363636 36363636

(Key ⊕ ipad) || text:
06070405 02030001 0e0f0c0d 0a0b0809
76777475 36363636 36363636 36363636
36363636 36363636 36363636 36363636
36363636 36363636 36363636 36363636
53616d70 6c652023 32800000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000248

Hash((Key ⊕ ipad) || text):
74766e5f 6913e8cb 6f7f108a 11298b15
010c353a

$K_0 \oplus$ opad:

6c6d6e6f 68696a6b 64656667 60616263
1c1d1e1f 5c5c5c5c 5c5c5c5c 5c5c5c5c
5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c
5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c

($K_0 \oplus$ opad) || Hash((Key $\oplus$ ipad) || text):

6c6d6e6f 68696a6b 64656667 60616263
1c1d1e1f 5c5c5c5c 5c5c5c5c 5c5c5c5c
5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c
5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c
74766e5f 6913e8cb 6f7f108a 11298b15
010c353a

HMAC(Key, Text) = Hash(($K_0 \oplus$ opad) || Hash((Key $\oplus$ ipad) || text)):

0922d340 5faa3d19 4f82a458 30737d5c
c6c75d24

20-byte HMAC(Key, Text):

0922d340 5faa3d19 4f82a458 30737d5c
c6c75d24

1.3 SHA-1 with 100-Byte Key

Text : "Sample #3"

Key : 50515253 54555657 58595a5b 5c5d5e5f
60616263 64656667 68696a6b 6c6d6e6f
70717273 74757677 78797a7b 7c7d7e7f
80818283 84858687 88898a8b 8c8d8e8f
90919293 94959697 98999a9b 9c9d9e9f
a0a1a2a3 a4a5a6a7 a8a9aaab acadaeaf
b0b1b2b3

Hash(Key):

a4aabe16 54e78da4 40d2a403 015636bf
4bb2f329

K_0 : a4aabe16 54e78da4 40d2a403 015636bf
4bb2f329 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000

$K_0 \oplus$ ipad:

929c8820 62d1bb92 76e49235 37600089
7d84c51f 36363636 36363636 36363636
36363636 36363636 36363636 36363636

36363636 36363636 36363636 36363636

(Key ⊕ ipad) || text:
929c8820 62d1bb92 76e49235 37600089
7d84c51f 36363636 36363636 36363636
36363636 36363636 36363636 36363636
36363636 36363636 36363636 36363636
53616d70 6c652023 33

Hash((Key ⊕ ipad) || text):
d98315c4 2152bea0 d057de97 84427676
2a1a5576

K_0 ⊕ opad:
f8f6e24a 08bbd1f8 1c8ef85f 5d0a6ae3
17eeaf75 5c5c5c5c 5c5c5c5c 5c5c5c5c
5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c
5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c

(K_0 ⊕ opad) || Hash((Key ⊕ ipad) || text):
f8f6e24a 08bbd1f8 1c8ef85f 5d0a6ae3
17eeaf75 5c5c5c5c 5c5c5c5c 5c5c5c5c
5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c
5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c
d98315c4 2152bea0 d057de97 84427676
2a1a5576

HMAC(Key, Text) = Hash((K_0 ⊕ opad) || Hash((Key ⊕ ipad) || text)):
bcf41eab 8bb2d802 f3d05caf 7cb092ec
f8d1a3aa

20-byte HMAC(Key, Text):
bcf41eab 8bb2d802 f3d05caf 7cb092ec
f8d1a3aa

1.4 SHA-1 with 49-Byte Key, Truncated to 12-Byte HMAC

Text : "Sample #4"

Key : 70717273 74757677 78797a7b 7c7d7e7f
80818283 84858687 88898a8b 8c8d8e8f
90919293 94959697 98999a9b 9c9d9e9f

K_0 : 70717273 74757677 78797a7b 7c7d7e7f
80818283 84858687 88898a8b 8c8d8e8f
90919293 94959697 98999a9b 9c9d9e9f
a0000000 00000000 00000000 00000000

$K_0 \oplus$ ipad:

46474445	42434041	4e4f4c4d	4a4b4849
b6b7b4b5	b2b3b0b1	bebfbcbd	babbb8b9
a6a7a4a5	a2a3a0a1	aeafacad	aaaba8a9
96363636	36363636	36363636	36363636

(Key $\oplus$ ipad) || text:

46474445	42434041	4e4f4c4d	4a4b4849
b6b7b4b5	b2b3b0b1	bebfbcbd	babbb8b9
a6a7a4a5	a2a3a0a1	aeafacad	aaaba8a9
96363636	36363636	36363636	36363636
53616d70	6c652023	34	

Hash((Key $\oplus$ ipad) || text):

bf1e889d	876c34b7	bef3496e	d998c8d1
16673a2e			

$K_0 \oplus$ opad:

2c2d2e2f	28292a2b	24252627	20212223
dcdddedf	d8d9dadb	d4d5d6d7	d0d1d2d3
cccdcecf	c8c9cacb	c4c5c6c7	c0c1c2c3
fc5c5c5c	5c5c5c5c	5c5c5c5c	5c5c5c5c

($K_0 \oplus$ opad) || Hash((Key $\oplus$ ipad) || text):

2c2d2e2f	28292a2b	24252627	20212223
dcdddedf	d8d9dadb	d4d5d6d7	d0d1d2d3
cccdcecf	c8c9cacb	c4c5c6c7	c0c1c2c3
fc5c5c5c	5c5c5c5c	5c5c5c5c	5c5c5c5c
bf1e889d	876c34b7	bef3496e	d998c8d1
16673a2e			

HMAC(Key, Text) = Hash(($K_0 \oplus$ opad) || Hash((Key $\oplus$ ipad) || text)):

9ea886ef	e268dbec	ce420c75	24df32e0
751a2a26			

12-byte HMAC(Key, Text):

9ea886ef	e268dbec	ce420c75

APPENDIX - 2 : SCHEMATIC DIAGRAMS OF OUR SCHEME

1. Detecting Unsolicited Messages and Anti-nodes

Sink determines the sensor nodes in the same region and the frequencies of the messages sent by them. If there is a sensor node generates δ times more messages than the other nodes where δ is a system parameter, this node is accepted as an anti-node. Otherwise, it is assumed that there is no anti-node in the network. This procedure is depicted in Figure 24.

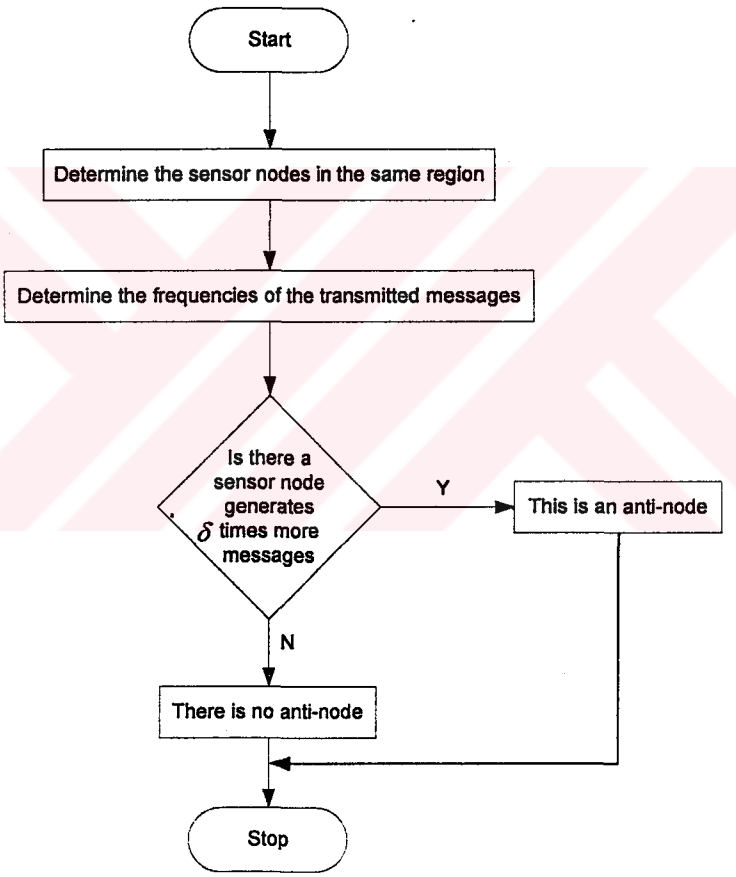

Figure 24. Detecting anti-nodes.

2. Determining the Borders of a Quarantine Region

Quarantine region is a square shaped region that has an anti-node at its center. The sink obtains the x and y coordinates of the anti-node from the source location field of the spam message. The distance d_q between the borders of a

quarantine region and the anti-node is given by $d_q = \beta \times d_{max}$ where β and d_{max} are system parameters. By using d_q, x and y coordinates of the anti-node, upper left u and lower right l corners of the quarantine region are found. This process is shown in Figure 25.

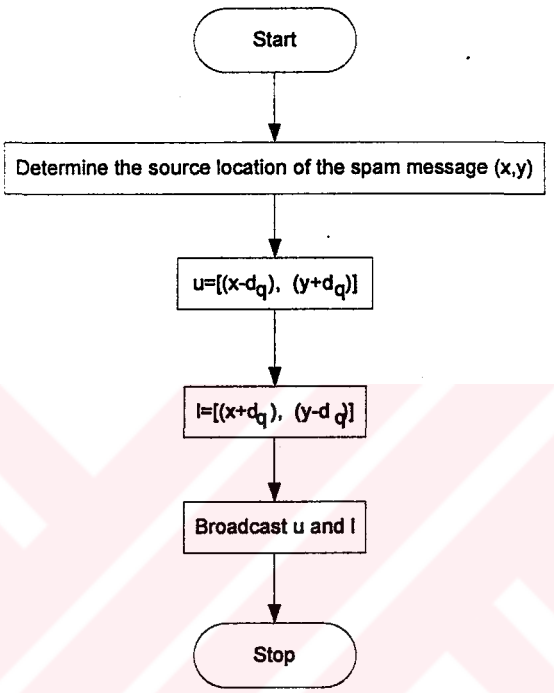

Figure 25. Determining the borders of a quarantine region.

3. Authentication in a Quarantine Region

When a sensor has message to send, it first checks its location. If it is in a quarantine region, it has to send messages authenticated. Therefore, it generates the *authentication code* of the message by using the key and HMAC algorithm. Then increase the *sequence number* and send the message with *authentication code*. Receiver takes the message and generates the *authentication code* again by using the same key and HMAC algorithm. The receiver compares the calculated value with the *authentication code* field of the incoming message. If they are not equal, the message is discarded. If they match, receiver compares the *sequence number* of the received message with the last sequence number which is obtained from each of its neighboring nodes. If the received message has a higher sequence

number, then it is concluded that the message is authentic. If the *sequence number* in the message is equal or less than the preceding ones, the receiver asks the *authentication code* for the same message but with expected *sequence number*. This procedure is depicted in Figure 26.

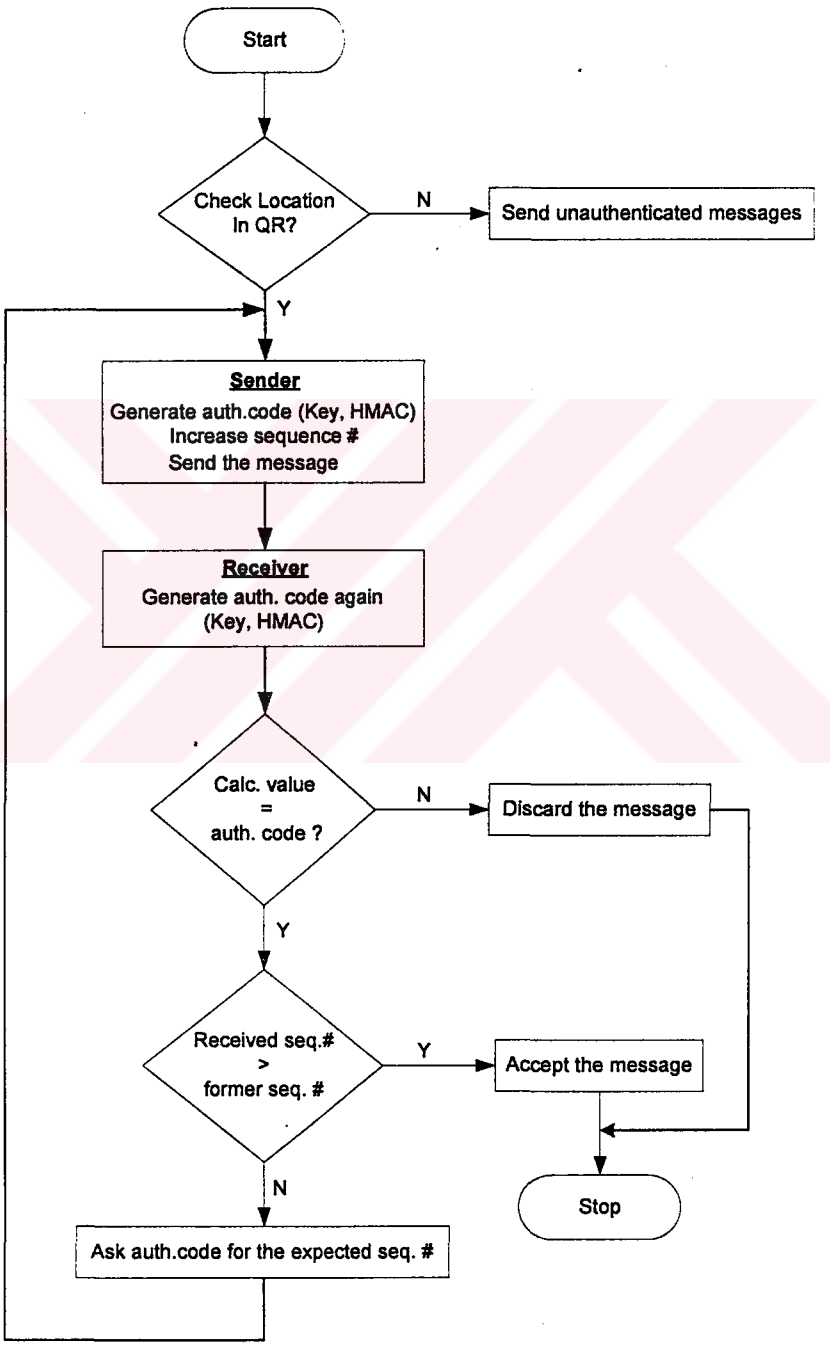

Figure 26. Authentication in a quarantine region.

4. Cancel Quarantine Region

If a sensor node receives an unauthenticated message from a quarantine region, it sets the *spam continue flag*. At the end of the *status report period* (t_q), if there is a flag set, each sensor node sends a spam report to the sink and removes the flag. Otherwise the sensor node does nothing. If the sink does not receive a spam report during the *status report period*, it cancels the quarantine region. Otherwise the *status report period* is start again. This procedure is shown in Figure 27.

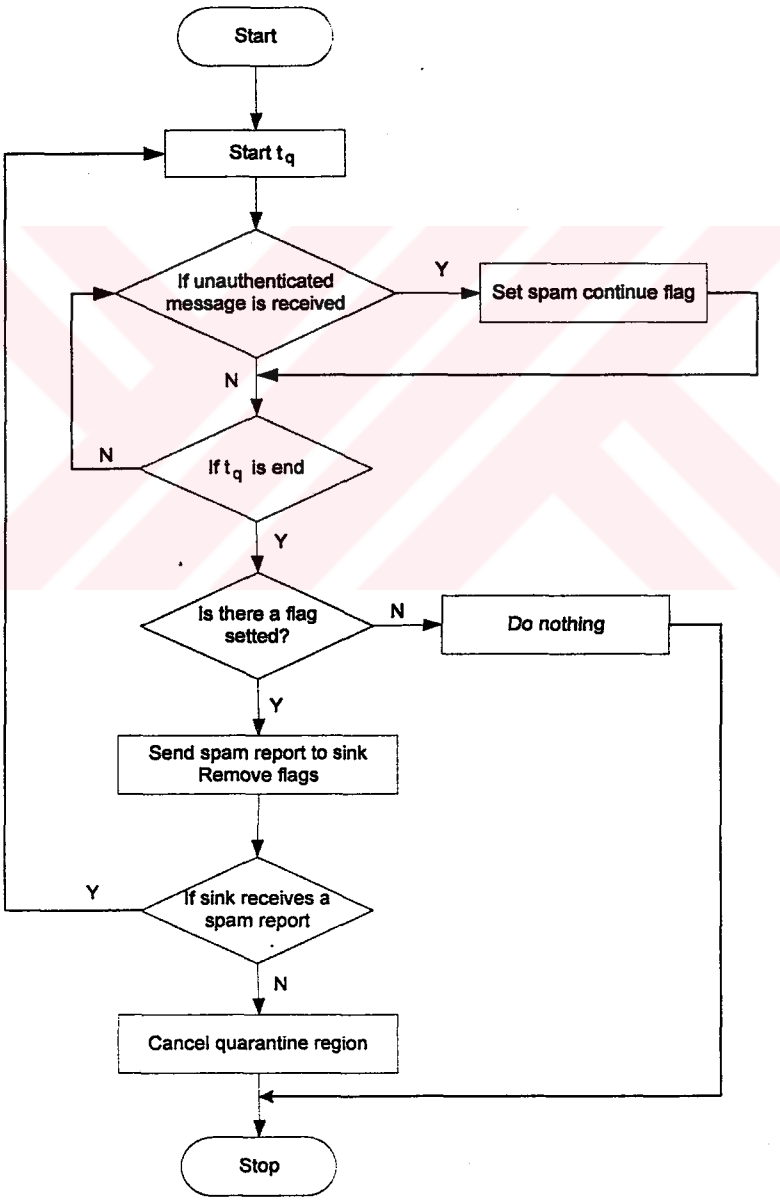

Figure 27. Cancel quarantine region.

APPENDIX -3 : PSEUDOCODE OF THE SIMULATION PROGRAM

```
% sensor_node : A matrix which contains sensor nodes
% transmit_range : transmit range of a sensor node
% num_of_sensor : number of the sensor nodes
% num_of_anti_sensor : number of the anti-nodes
% tr_range_table : A matrix which contains the neighbour nodes of each
%sensor node
% quarantine_region : A matrix which contains the nodes in the
%quarantine region
% path : A matrix which contains the path to the sink for each sensor node
% anti_node : A matrix which contains the hop counts of caused by each
%anti_node

Sensor_War( )  % Main program
{
  deploy_sensor_node_random ( )
  sensor_node=arrange(sensor_node)
  plot sensor nodes

%Finding the neighbours and the distances all of the nodes starting from
%sink within the transmit range

  for counter=1:num_of_sensor,
  {
    for i=1:num_of_sensor,
    {
      distance = distance of sensor_node(i) from sensor_node(counter);
      if transmit_range>=distance
        tr_range_table(counter)=sensor_node(i) %#of node and distance;
    }
```

```
}
for counter=1:num_of_sensor,
{
  for i=1:num_of_sensor,
  {
    if (sensor_node(i) is in the transmit_range of sensor_node(counter))
and (there is no path to sensor_node(i))
      path(counter)=sensor_node(i);
  }
}

deploy_anti_node( )
teta=1;
dmax=transmit_range*teta;
quarantine_region=sensors_in_tr_range(anti_node,sensor_node,dmax);
total=0;

for i=1:number of anti_node,
{
  for j=1:number of sensors affected by the anti_node(i),
  {
    if not reached to the sink
      total=total+hop_count(path,quarantine_region(j,i));
  }
  anti_hop(i)=total;
}
plot anti_node;
}
```

```
function sensors_in_tr_range=sensors(anti_node,sensor_node,
transmit_range)
{
% Finding the sensor-nodes which are affected by each of the anti-node
%and put them in quarantine region

  for counter=1:num_of_anti_sensor,
  {
% Finding the neighbours of anti_nodes within the transmit range
    for i=1:num_of_sensor,
    {
      distance = distance of anti_node(counter) from sensor_node(i);
      if transmit_range>=distance;
      quarantine_region=sensor_node(i);
    }
  }

%Finding the upper-left and lower-right coordinates of quarantine region
  if quarantine region is empty
    display("There is no sensor node which affected by any anti-node, so
    quarantine region is not constituted");
  else
  {
    Xmax = find the biggest X-coordinate in quarantine region ;
    Xmin = find the lowest X-coordinate in quarantine region ;
    Ymax = find the biggest Y-coordinate in quarantine region ;
    Ymin = find the lowest Y-coordinate in quarantine region ;
    upper-left = Xmin, Ymax ;
    lower-right = Xmax, Ymin ;
  }
}
```

```
function arrange=matris(sensor_node)
{
  for counter=1:num_of_sensor,
  {
    distance = distance of sensor_node(counter) from sink;
    sensor_node(counter)=distance;
  }

%We use "bubble sort" algorithm to arrange the sensor nodes according to
%their distance.
  change=1;
  while(change~=0)
    change=0;
    for i=1:num_of_sensor,
    {
      if sensor_node(i)> sensor_node(i+1)
      {
        change=1;
        x= sensor_node(i);
        sensor_node(i)= sensor_node(i+1);
        sensor_node(i)=x;
      } %if end
    } % for end
  } % while end
} %function end
```

```
function hop_count=matris(path,n)   %n=sensor_node_number
{
  location=n;
  hop=0;
  degis=0;
  while((location~=sink)&(sensor_node(n) has a path)),
  {
    hop=hop+1;
  }
```

% (sensor_node(n) has a path) condition protects us from a vicious circle. Because

% if there is any sensor does not have a path to sink, that sensor never reach to sink.