BABRA: Batch-based Broadcast Authentication in Wireless Sensor Networks

Yun Zhou, and Yuguang Fang
Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611
Tel: (352)392-8576; Fax: (352)392-0044; Email: {yzufl@, fang@ece.}ufl.edu

***Abstract*— To prevent adversaries from injecting bogus messages, authentication is required for broadcast in wireless sensor networks. μTESLA is a light-weight broadcast authentication protocol, which uses a one-way hash chain and the delayed disclosure of keys to provide the authentication service. However, it suffers from several drawbacks in terms of time synchronization, limited broadcast rounds, key chain management at the source node, etc. In this paper, we propose a novel protocol, called *BAtch-based BRoadcast Authentication* (BABRA) for wireless sensor networks. BABRA does not require time synchronization, eliminates the requirement of key chain, and supports broadcast for infinite rounds. Like μTESLA, BABRA is also efficient due to the use of symmetric key techniques.**

I. Introduction

Wireless Sensor Networks (WSNs) [1] have been drawing a lot of interests because of their wide applications such as event detection and target tracking in both civil and military scenarios. Usually, a WSN consists of hundreds or even thousands cheap sensor nodes, which collaborate with each other and communicate with external world through one or several powerful nodes, called base stations.

Broadcast is a common communication pattern to fulfill collaboration among sensor nodes. For example, the base station may spread messages such as commands or requests to the entire network through the *network broadcast*. Each individual node may use the *local broadcast* to fulfill some specific functions in its neighborhood, such as exchanging routing information or cluster head election. Therefore, the correct broadcast is critical to the collaboration objective of sensor networks. In hostile environments, however, adversaries may take the advantage of broadcast to inject false information, which can raise significant havoc in the network. To defeat such an attack, authentication is required. Each broadcasted packet should carry some authentication information so that the recipient node can verify its authenticity.

μTESLA [2] is a light-weight broadcast authentication protocol, which uses a one-way hash key chain and the delayed disclosure of keys to provide the authentication service. It is efficient due to the use of symmetric key techniques. However, it requires synchronization between the source and recipient, which can be a potential security hole for adversaries [3]. Moreover, the key chain in μTESLA has limited length, and thus can only support limited rounds of broadcast. If the source node needs to broadcast for a long period, it has to generate a long key chain. But the management of a long key chain is difficult for low-end sensor nodes. So μTESLA can only be used by the base stations for the network broadcast.

This work was supported in part by the US Office of Naval Research under grant N000140210464 (Young Investigator Award) and under grant N000140210554, and by the US National Science Foundation under grant ANI-0093241 (CAREER Award) and under grant ANI-0220287.

In this paper, we propose a novel protocol, called *BAtch-based BRoadcast Authentication* (BABRA) for wireless sensor networks. BABRA broadcasts packets in batches and the transmissions of different batches do not require time synchronization. Therefore, BABRA eliminates the security hole that μTESLA suffers. Moreover, BABRA uses independent keys instead of a key chain for different batches, and thus supports broadcast for infinite rounds. In addition, BABRA is also built on symmetric key techniques and thus is efficient.

The rest of the paper is organized as follows. Section II simply describes the μTESLA protocol. Details of BABRA are given in Section III. Some comparisons between μTESLA and BABRA are carried out in Section IV. The paper is finally ended in Section V.

II. μTESLA

Though public key signatures can provide authentication services, they are too expensive for sensor networks. Therefore most researchers are seeking symmetric key solutions. μTESLA is a broadcast authentication protocol, which is a simplified version of TESLA [4]. It is based on a *one-way hash chain* (OHC), which is a sequence of keys, $K_0, K_1, \ldots, K_n$, such that $K_{j-1} = H(K_j)$, $\forall j$, $j > 0$, where the hash function H satisfies two properties:

1) Given x, it is easy to computer $y = H(x)$;
2) Given y, it is computationally infeasible to compute x such that $y = H(x)$.

The first key K_0 is unicasted to all the recipient nodes as a commitment in advance. The entire broadcast stream is divided into continuous time slots. A broadcasted packet in the t-th time slot carries a *Message Authentication Code* (MAC) generated by using the t-th key K_t of the OHC. All the recipient nodes dos not know K_t when they receive the packet. After d time slots, the source node discloses K_t. Then every node can authenticate K_t by applying the hash function to K_t several times and checking whether $H^k(K_t) = K_{t-k}$ holds, where K_{t-k} is the $t-k$-th key that has been received and authenticated. After that, the recipient node can use the authenticated K_t to authenticate the packets of the t-th slot. The delayed key release can efficient prevent malicious nodes

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.

from impersonating the source node, because the disclosed key K_t can not be used to spoof packets after the t-th slot.

Though μTESLA is more efficient compared with other public key signatures protocols, it has some strong requirements. The slot-based broadcast requires time synchronization throughout the entire network. The time synchronization procedure may undergo potential threats leading to the failure of the entire protocol [3]. The distribution of the key chain commitment K_0 to all the nodes is communication expensive because the commitment has to be unicasted to each node while the network can consist of large volume of nodes. The OHC length is limited, and thus it can not support broadcast for a long time. The complex key chain management indicates that μTESLA can be used only by the base station for the network broadcast.

Multilevel key chains are used to extend the lifetime of authenticated broadcast [5], but it is still limited by the highest level OHC. The multilevel key chains also require the source node manage many OHCs at the same time and thus are not suitable for sensor nodes. Moreover, time synchronization is still a requirement.

III. BABRA Design

Unlike μTESLA, BABRA do not require time synchronization, and supports broadcast for infinite rounds. It can be used in both the network broadcast and the local broadcast. In this section, we give the details of BABRA.

A. Network Model

We consider the application scenarios including the network broadcast, where the base station broadcasts messages into the entire network, and the local broadcast, where each node broadcasts messages in its one-hop neighborhood. BABRA can provide authentication services for these broadcast patterns. Though confidentiality is also critical to group communications, the management of encryption keys is a very challenging task [6] and is out of our considerations.

The main purpose of authenticating broadcast is to prevent adversaries from injecting bogus packets. BABRA also uses delayed key disclosure to counteract the bogus packet injection. In addition, adversaries can also inject radio interference at the physical layer to disrupt communications, leading to the DoS attack [7]. The intermittent interference can deteriorate channel condition and cause packet loss. The continuous jamming can even stop communications. However, due to the large scale of network and cost considerations, adversaries may not be able to jam the entire network. In this paper, we assume that the impact of radio jamming only covers a portion of the network at one time.

There are other attacks and corresponding countermeasures discussed in the literature [7], [8]. They are out of the scope of this paper because most of them are unrelated to broadcast authentication. We have developed several schemes [9]–[11] to establish pairwise keys to secure point-to-point communications. In this paper, we simply assume that every pair of neighboring nodes shares a pairwise key after network initialization.

B. Architecture

In BABRA, broadcasted packets are sent in *batches* and each batch is a burst sequence of packets. There is a key associated with each batch. All the packets in one batch carry an MAC calculated based on the associated key, and are sent in C time units, which is the *batch period* (BP). At the end of the BP, the source node starts a timer of D units, which is the *delay period* (DP). During the BP and the DP, the batch key is kept secret by the source node. When the DP timer expires, the source node discloses the corresponding batch key in a *key disclose period* (KP) of E time units. When a recipient node gets the first packet of the batch, it starts a timer last for C time units. Only the batch packets that arrive within the period of C time units are accepted by the recipient. At the end of the period, the recipient starts a new timer for D time units as the DP. After the DP, the recipient can receive the corresponding batch key and use the key to recalculate the MAC to check the authenticity of the cached batch packets. Due to the delayed key disclosure, the adversary can not use the disclosed key to inject bogus batch packets because the source node never sends any packet of this batch after the key disclosure period.

However, each batch key should be authenticated before being used to authenticate the corresponding batch packets. BABRA achieves this goal by using an immediate authentication method proposed in [12]. Particularly, all the packets in one batch also carry a hash of the key associated with the next batch. Hence, each broadcasted packet consists of four parts: the batch index, the payload, the hash of the key of next batch, and the MAC calculated over the previous three parts and the batch key (Fig. 1). The delayed batch key can authenticate the corresponding batch. The hash of the key of next batch is authenticated at the same time, and can be used to authenticate the key of next batch.

The entire broadcast stream is depicted in Fig. 2. Before broadcast, the source node bootstraps all the recipient nodes with the hash $H(K_1)$ of the first batch key K_1. Depending on the scenarios where BABRA is applied, different methods can be used to bootstrap the hash value. We will discuss this issue later. After bootstrapping, the source node can send out batches of broadcasted packets one by one and disclose the corresponding batch keys lately (Fig. 2). Each batch is not necessary to be sent right after the end of the previous batch. Therefore, BABRA can be adapted to different data rates.

In hostile environments, the adversary can inject jamming interference and cause packet loss. The nodes in the jammed area will seek help from the surrounding neighbors to recover the lost information such as keys or key hashes. To facilitate such local collaboration, each recipient node keeps the latest k keys received from the source node. We will discuss this issue in Section III-E.

C. Bootstrapping

As is mentioned before, the hash $H(K_1)$ of the first batch key K_1 needs to be bootstrapped into all the recipient nodes. To avoid using expensive public key signatures to authenticate

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.

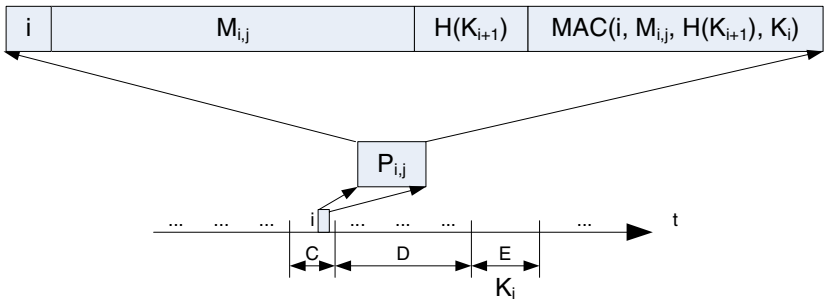

Fig. 1. One batch of broadcast and the batch packet format. All the packets in one batch are sent in a batch period of C time units. At the end of the batch period, the source node delays for a delay period of D time units and then discloses the corresponding batch key in a key disclose period of E time units. All the packets in one batch carry an MAC calculated based on the associated batch key, and include a hash of the key associated with the next batch.

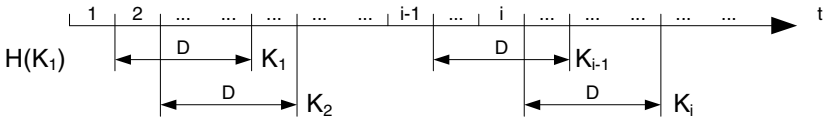

Fig. 2. The authenticated broadcasting stream. The source node sends out batches of broadcasted packets one by one and discloses the corresponding batch keys lately.

$H(K_1)$, we need some methods based on symmetric key techniques.

For the local broadcast, the source node can unicast $H(K_1)$ to each of its neighbors. Each unicast is authenticated with the pairwise key shared between the source and the corresponding neighbor. Because the number of neighbors is small, such unicast bootstrapping can be finished in a very short time period.

Though unicast can also be used to bootstrap the network broadcast, the overhead is too much because there are too many nodes in a network. It takes too much time for the base station to unicast to each node. A simple way to bootstrap the network broadcast is to preload each node with $H(K_1)$ before deployment. It is easy to achieve this because the entire sensor network is usually managed under a unique authority, and thus preloading secure parameters is a common way to establish a secure architecture for the sensor network [2], [5], [9]–[11].

D. Counteracting Bogus Packets

The parameter DP is critical to the security of the entire broadcasting protocol. If the value of DP is small, there is a chance that the adversary catches the key before some nodes get the corresponding batch packets and then sends bogus packets towards these nodes. Therefore, the value of DP should be large enough for all nodes to get the batch before the release of the key. For the local broadcast, the DP value D_l is larger than maximum one-hop transmission delay, and thus can be set as

$$D_l = \lambda_l \left(\frac{R}{c} + P \right) , \tag{1}$$

where $\lambda_l > 1$ is a constant, R is the radius of node coverage, c is the speed of light, and P is the packet processing delay. For the network broadcast, the DP value D_n should be larger than the time that a packet is transmitted over the maximum diameter L of the network, and thus can be set as

$$D_n = \lambda_n \frac{L}{R} \left(\frac{R}{c} + P \right) , \tag{2}$$

where $\lambda_n > 1$ is a constant.

E. Countermeasures to Radio Jamming

The adversary can introduce jamming interference to disrupt communications, leading to the DoS attack. The intermittent interference can deteriorate channel condition and cause packet loss. The continuous jamming can even stop communications. Here we discuss their impacts and countermeasures.

1) Intermittent Jamming: Each batch of broadcasting is authenticated by the corresponding batch key. If some of the batch packets are lost due to jamming, the recipient just experiences lower quality of service. But if the batch key is lost, the entire batch is useless. Therefore, to tolerate the key loss is a very important task. Here we introduce the following two methods to solve this problem.

To provide resilience to the key loss, the first method in BABRA is to transmit each batch key several times during the corresponding key disclose period. Suppose the average packet loss rate is p_l, and each batch key is transmitted t times during its KP. The probability that the key can be received is

$$P = 1 - {p_l}^t . \tag{3}$$

Fig. 3 gives the key survival probabilities P versus the key disclose times t when the packet loss rate p_l varies from 0.2 to 0.8. We can see by simply disclosing multiple times, the batch key can be received with very high probability.

It is worth noting that to disclose key multiple times is the simplest *Forward Error Correction* (FEC) method to counteract packet loss in communications. More complex and

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.

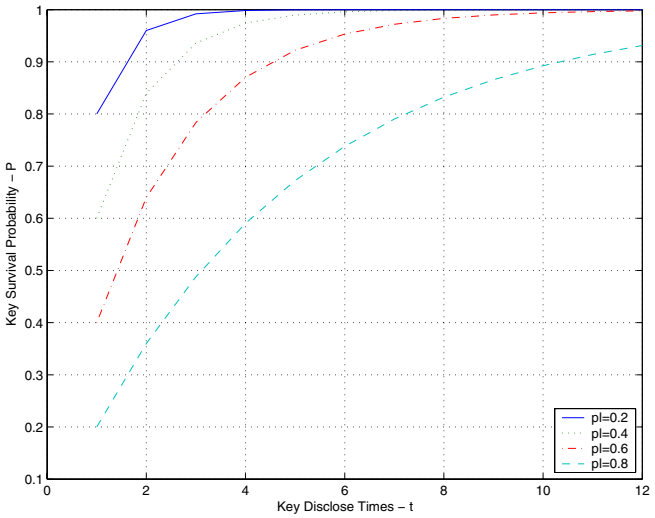

Fig. 3. The key survival probability P versus the key disclose times t, with respect to different packet loss rate p_l.

robust FEC methods can also be used here to increase the resilience to the key loss. One example is to use *Reed-Solomon* codes [13]. We do not discuss this issue here for the sake of page limit.

The second method is carried out just in case that it is unlucky that all the t receptions of batch key fail. In such a case, the recipient node will seek help from its neighbors right after the expiration of the KP timer. For the key loss during the network broadcast, the node will locally broadcast a message to request its neighbors for the lost key:

$$a \longrightarrow *: \ \langle j, H(K^a_{i+1}), MAC(j, H(K^a_{i+1}, K^a_i)\rangle \ ,$$

where j is the index of the batch of which the key is lost, K^a_{i+1} is the key associated with the next batch of node a's local broadcast stream, and K^a_i is the key of the current batch of a's local broadcast stream. Therefore this message is authenticated by the local broadcast authentication. The adversary can not spoof the message.

If a neighbor node b knows the key K_j, it will reply a message through a local broadcast message:

$$b \longrightarrow *: \ \langle K_j, H(K^b_{i+1}), MAC(K_j, H(K^b_{i+1}, K^b_i)\rangle \ ,$$

where K^b_{i+1} is the key associated with the next batch of node b's local broadcast stream, and K^b_i is the key of the current batch of b's local broadcast stream. This message is also authenticated so that it can not be spoofed. In addition, the key K_j is broadcasted, so node b has no bonus of replying bogus messages because nearby nodes that also have K_j can check whether node b lies to node a. This local monitoring has been used in misbehavior detection. One example is discussed in [14].

When node a gets K_j from its neighbor b, it will broadcast K_j again through its local authenticated broadcast so that its neighbors know that it really gets K_j.

If none of node a's neighbors knows K_j, they will continue the above procedure until some node can reply with K_j. For example, if node b does not get K_j but its neighbor c knows K_j, then b can learn K_j from c. Then b can broadcast K_j if it has a request from a.

If multiple nodes in the neighborhood of node a knows K_j, all of them might try to reply at the same time. But the underlying contention resolving mechanisms at the Media Access Layer can guarantee that one of them replies successfully. Other nodes that hear the replying of K_j stop trying to broadcast K_j.

For the key lost during the local broadcast, it is easy to resend the key from the source node to the recipient node because it only involves one-hop communication, which can be encrypted and authenticated by the pairwise key shared between the source and the recipient.

2) Continuous Jamming: Continuous jamming is more severe to broadcast. When the channel is jammed, the recipient gets nothing. Because the key of each batch is authenticated by its hash included in the previous batch, the key can not be authenticated if all the packets of the previous batch are lost due to the continuous jamming. Here we need some measures to help recipient nodes recover the interrupted broadcast stream when the jamming attack stops.

When the recipient node a gets a packet in the next batch right after the jamming attack, node a broadcasts a message including the index, say j, of the batch and the index i of the last batch it receives just before the jamming attack. This message is authenticated by node a's local broadcast protocol, i.e.,

$$a \longrightarrow *: \ \langle j, i, H(K^a_{l+1}), MAC(j, i, H(K^a_{l+1}, K^a_l)\rangle \ ,$$

where K^a_{l+1} is the key associated with the next batch of node a's local broadcast stream, and K^a_l is the key of the current batch of a's local broadcast stream. Node a broadcasts the index j for the hash $H(K_j)$ of the key K_j associated with the batch right after the jamming attack. In addition, the packets of several batches just before the jamming attack are cached in a's buffer and not authenticated due to the loss of the corresponding keys during the jamming attack. So node a also broadcast the index i of the last batch just before the jamming attack. Because every node will cache the latest k keys disclosed by the source, if any neighbor finds in its buffer that there are keys associated with some batches before the jamming attack, it can reply with those keys so that node a can authenticate the packets of those batches.

When a nearby node b, which is uninfluenced from the jamming attack and has cached k latest keys $K_m, \ldots, K_{m-k+1}$, replies with an authenticated broadcast message as:

$$b \longrightarrow *: \ \langle H(K_j), [K_i, \ldots, K_{m-k+1}], H(K^b_{l+1}),$$
$$MAC(H(K_j), [K_i, \ldots, K_{m-k+1}], H(K^b_{l+1}, K^b_l)\rangle \ ,$$

where K^b_{l+1} is the key associated with the next batch of node b's local broadcast stream, and K^b_l is the key of the current batch of b's local broadcast stream. $H(K_j)$ is used to authenticate the key K_j of the next batch j right after the jamming attack, and then the key K_j is used to authenticate the packets

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.

in the batch. Here the keys $K_i, ..., K_{m-k+1}$ are optional. Node b checks the latest k cached keys $K_m, \ldots, K_{m-k+1}$. If the index $i \geq m - k + 1$, node b knows that node a needs the keys from K_{m-k+1} to K_i to authenticate the packets of the last several batches just before the jamming attack. Then node b replies with these keys.

Due to the broadcast, the surrounding nodes that also have copies of $H(K_j)$ can check whether node b lies to node a. Hence node a can get correct $H(K_j)$ and use it to authenticate K_j later whereby to recover the entire broadcast stream. When node a gets $H(K_j)$ and/or $K_i, ..., K_{m-k+1}$ from its neighbor b, it will broadcast them again through its local authenticated broadcast so that its neighbors know that it really gets them.

If all the neighbors of node a do not know the information that a desires, they will continue the above procedure until there is at least one node can give those information.

For the local broadcast under the jamming attack, it is easy for the recipient node to recover after the jamming. Through unicast, the recipient node a can get all the required information from the source node b, where the communication is encrypted and authenticated by the pairwise key shared between a and b.

IV. Disscussion

Each packet in BABRA and that in μTESLA both carry an MAC as the authentication information. The difference is that BABRA replaces the timestamp in the μTESLA packet with a batch index and a key hash. The batch index is just like the timestamp and thus can be represented with the same number of bits. However, BABRA does not use limited key chain and not require each batch to be sent right after the end of the previous batch, so BABRA can support a longer lifetime of a broadcast stream. Suppose each batch period be $100ms$, which is corresponding to one time slot [5]. A 32-bit batch index can support a broadcast stream up to 4971 days if the source keeps sending batches continuously.

As for the key hash in each BABRA packet, its length should guarantee that no two keys have the same hash value. Otherwise, the adversary can spoof broadcasted packets. Considering the 32-bit batch index, the number of keys in BABRA is 2^{32}. According to the *birthday paradox* [15], a 64-bit key hash is enough to guarantee that all the 2^{32} keys generate different hash values with a probability close to 1. Though BABRA introduces the additional packet overhead for the key hash, it is worth because of the elimination of the time synchronization requirement.

Like μTESLA, BABRA also requires every node to buffer packets before the corresponding key is disclosed. The difference is the management of keys. In μTESLA, the source node has to manage a key chain, which has a length determined by the lifetime of the broadcast stream. However, to manage such a key chain may not be feasible when the source wants to broadcast for a long time. In BABRA, all the keys are independent. The elimination of key chains makes BABRA suitable for both the network broadcast by base station and the local broadcast by sensor nodes. Each node in BABRA caches the latest k keys disclosed by the source node. The value k can be adapted according to the buffer space of each node.

V. Conclusion

Though there are many broadcast authentication protocols proposed for conventional wired networks, few work has been carried out for wireless sensor networks. Though μTESLA can provide the broadcast authentication service for sensor networks, it still suffers some drawbacks. BABRA is a batch-based broadcast authentication protocol for wireless sensor networks. BABRA broadcasts packets in batches and the transmissions of different batches do not require time synchronization. Therefore BABRA eliminates the security hole that μTESLA suffers. BABRA uses independent keys in stead of a key chain for different batches, and thus supports broadcast for infinite rounds. BABRA can support both the network broadcast and the local broadcast. In addition, BABRA is also built on symmetric key techniques and thus is efficient.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communication Magazine*, vol. 40, no. 8, pp. 102-114, August 2002.

[2] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: Security protocols for sensor networks," Wireless Networks 8, 521-534, 2002.

[3] M. Manzo, T. Roosta, and S. Sastry, "Time synchronization attacks in sensor networks," *ACM SASN'05*, Alexandria, Virginia, November 7, 2005.

[4] A. Perrig, R. Canetti, B. Brisco, D. Song, and D. Tygar, "TESLA: Multicast source authentication transform introduction," *IETF* working draft, draft-ietf-msec-tesla-intro-01.txt.

[5] D. Liu and P. Ning, "Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks," *NDSS'03*, 2003.

[6] S. Rafaeli and D. Hutchison, "A survey of key management for secure group communication," *ACM Computing Surveys*, Vol. 35, No. 3, September 2003.

[7] A. Wood and J. Stankovic, "Denial of service in sensor networks," IEEE Computer, pp. 54-62, October 2002.

[8] C. Karlof, D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," First IEEE International Workshop on Sensor Network Protocols and Applications, May 2003.

[9] Y. Zhou, Y. Zhang, and Y. Fang, "LLK: A link-layer key establishment scheme in wireless sensor networks," *IEEE WCNC'05*, New Orleans, LA, March 2005.

[10] Y. Zhou, Y. Zhang, and Y. Fang, "Key establishment in sensor networks based on triangle grid deployment model," to appear in *IEEE MILCOM'05*, Atlantic City, New Jersey, October 17-20, 2005.

[11] Y. Zhou and Y. Fang, "A scalable key agreement scheme for large scale networks," *2006 IEEE International Conference on Networking, Sensing and Control (ICNSC'06)*, Ft. Lauderdale, Florida, USA, April 23-25,2006.

[12] A. Perrig, R. Canetti, D. Song, and J.D. Tygar, "Efficient and secure source authentication for multicast," *NDSS'01*, 2001.

[13] C.A. Gunter, S. Khanna, K. Tan, and S. Venkatesh, "DoS protection for reliably authenticated broadcast," *Network and Distributed System Security* (NDSS'04), San Diego, CA, February 2004.

[14] I. Khalil, S. Bagchi and C. Nita-Rotaru, " DICAS: detection, diagnosis and isolation of control attacks in sensor networks," *IEEE Conference on Security and Privacy for Emerging Areas in Communication Networks* (SecureComm'05), Sept. 2005.

[15] C. Kaufman, R. Perlman, and M. Speciner. *Network Security: private communication in a public world*, 2nd Edition, Prentice-Hall.

1-4244-0357-X/06/$20.00 ©2006 IEEE

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.