Intrusion Detection in Wireless Sensor Networks

Michael Krishnan

Abstract

Wireless Sensor Networks (WSNs) offer an excellent opportunity to monitor environments, and have a lot of interesting applications in warfare. The problem is that security mechanisms used for wired networks do not transfer directly to sensor networks. Some of this is due to the fact that there is not a person controlling each of the nodes, and even more importantly, energy is a scarce resource. Batteries have a short lifetime and cannot be replaced on deployed sensor nodes. In this paper I look at some of the special actions that need to be taken in WSNs versus wire-line networks, reviewing some of the approaches to intrusion detection as well as offering a new game theoretic-approach.

I. Introduction

Wireless sensor networks (WSNs) have become a hot research topic in recent years. Applications include military, rescue, environment monitoring, and smart homes. A WSN is composed of hundreds or even thousands of small, cheap sensors nodes which communicate with one another wirelessly. Sensor nodes typically do not have very much computational power, limiting the kinds of networking protocols and security mechanisms they can employ. Because WSNs are composed of so many nodes, which may be deployed in a hostile environment, replacing batteries is not feasible. Sensor nodes must therefor survive on the small amount of energy in the batteries they are deployed with (typically about 6 amp-hours [5]). This creates a need to conserve energy. Because of the wireless nature of WSNs, security is a fairly difficult issue. Adversaries can easily listen to all the traffic and inject their own, especially if the WSN is deployed in a hostile environment. It is also important that the WSN be robust to losing some of the sensor nodes, because it can be very easy for an adversary to capture any given node.

The general network topology is a dense collection of nodes, randomly distributed over some geographic area. Traffic typically goes from all the sensor nodes to a single sink, called the base station (BS), or broadcast traffic. A lot of work is currently being done on routing protocols, and not all of the details are figured out and agreed upon but, in general, routing is multi-hop like an ad-hoc wireless network. Cluster-based routing is a popular idea, because it is possible to exploit the fact that nearby nodes have highly correlated data [10]. In cluster-based routing, the network is divided up into clusters, which consist of a cluster head (CH) and member nodes (MNs). The MNs send their data to the CH, which aggregates the data before sending it out of the cluster toward the base station.

In this paper, I will assume that the WSN is in a hostile environment, where the deployers have no physical contact with the nodes, but attackers may. An attacker is someone who tries to disturb the functionality of the network in any of the ways described in II. I will also talk about "intrusion detection". This is defined as identifying an intruder, which is an attacker who has gained control of a node, or injected falsified or repeated packets into the network. This is not to be confused with other "intrusion detection" systems *using* WSNs, which monitor a physical environment, looking for intruders using a WSN for sensing and collecting information.

In this paper, I will explain the special circumstances presented by WSNs, summarize work that has been done in security for WSNs and propose a new game theory-based approach for intrusion detection. In section II, I will classify the types of attacks on WSNs. In section III, I will explain a few of the protocols that can be used for WSN security. In section IV, I will discuss how clusters can be used for security purposes. In section V, I will present a game theoretic approach to intrusion detection, and in section VI I will discuss my modified version of this game. Finally, I will conclude the paper in section VII.

II. Classification of Attacks

There are four aspects of a wireless sensor network that security must protect:

1) Confidentiality
2) Data Integrity
3) Service Availability
4) Energy

The first three are addressed by security systems in wired networks and non-energy-constrained wireless networks, but the fourth is unique to the sensor network application.

I will thus classify the attacks that can be launched by which of these aspects they attack. In this section I will briefly describe the kinds of attacks in each of these categories and the ways in which the nature of the wireless network causes trouble.

A. Stealing Data (Confidentiality)

When we think of electronic security, this is the first kind of attack that comes to mind. We want to be able to send messages without enemies being able to figure out the contents. Because of the wireless nature of the WSN, it is easy for an attacker to listen in on all the messages sent in the network, so to maintain confidentiality, the network must encrypt all the messages.

One of the biggest ideas in encryption today is public-key encryption. This is very powerful because it allows one to receive encrypted messages without even sharing a secret key with the sender. But this asymmetry comes at a cost. RSA public key encryption involves exponentiating the message, which can be quite computationally expensive, and is not really feasible in WSNs, where the nodes typically are not capable of doing such computations in a time- and energy-efficient way. But symmetric key encryption mechanisms pose the problem that if any node is apprehended by the attacker, he can look in its memory to find the key and effectively be able to masquerade as any other node (compromising the data integrity) and listen in on any other conversation.

B. Altering/Generating False Data (Data Integrity)

Because sensor networks are used to monitor some environment, data integrity is even more important than confidentiality. This is because applications may include tracking objects that physically move through the environment and since attackers can typically see the physical environment, the only thing they could gain from listening in on the data is a sense of where the sensors are located. On the other hand, if they are able to alter to make the data collected by the WSN incomplete or incorrect, the deployer of the WSN will not know what is really going on in the environment he is trying to monitor.

In other networks, the same asymmetric key system that is used for encryption can be used for digital signatures, but this requires a lot of additional overhead. The signature may consist of a lot of additional bytes of data added on to a transmission (which takes additional energy), and verifying the signature can be very computationally expensive. Clearly, different techniques are needed for WSNs.

C. Attacks on Service Availability

This class of attacks is not at all concerned with the actual data that is begin sent. Rather, the goal is to make the network not function properly. This can be done by sending bogus routing information (for example advertising a route that does not exist). It can also be done by flooding the network with packets (denial of service attack), or even jamming the frequency at the physical layer.

Another interesting type of attack is homing. In a homing attack, the attacker looks at network traffic to deduce the geographic location of critical nodes, such as cluster heads or neighbors of the base station. The attacker can then physically disable these nodes. This leads to another type of attack: the “black hole attack”. In a “black hole” attack, the attacker compromises all the neighbors of the base station, making it effectively a black hole. A final kind of attack on service availability is a de-synchronization attack, where the attacker tries to disrupt a transport-layer connection, by forging packets from either side [2].

D. Denial of Sleep Attacks (Energy)

The constrained energy of WSNs adds a new element that can greatly complicate security issues. Because there is a limited amount of energy available and no way to replenish it, it is not sufficient to make sure that bad data is not used. We need to make sure that we do not waste energy listening to or re-transmitting bad packets. This introduces a whole new set of possible attacks. These include constantly sending RTS packets to stop nodes from going to a low power “sleep” state, sending falsified or repeated packets so that nodes waste energy re-transmitting them, or draining the power of a node by forcing it to do excessive computations [5].

III. Soutions

A. SPINS

Many of the confidentiality and data integrity issues can be handled by SPINS [7]. SPINS is a collection of protocols for sensor networks. The key security components are SNEP and μTESLA.

SNEP provides a lot of key security features. It provides confidentiality and data integrity for pairwise connections as well as weak freshness. Freshness means that old packets cannot be repeated by an adversary to create confusion and waste energy. Weak freshness means that there are no delay guarantees, but packets cannot be repeated or re-ordered.

In SNEP, each pair of nodes shares a pair-wise key κ. This key is used in DES in cipher block chaining (CBC) mode. The cipher block chain provides semantic security (meaning that the same message string will not always encrypt to the same cipher string) through the use of an initialization vector (IV). Rather than sending this IV in the clear along with a message, the IV comes from a shared counter. This alleviates the need to send unnecessary bits. The counter also provides data freshness, because since it is incremented with each transmission, a previous transmission cannot be repeated, and the correct ordering is evident.

In addition to being encrypted, messages are also authenticated in SNEP through the use of a message authentication code (MAC), which is a function that takes two arguments and maps them to an 8 byte number. The arguments used are the pairwise key and the concatenation of the count and the encrypted message. Because all of these are available to the receiver, it can calculate the function to verify the signature.

SNEP is very good because it provides all of this security for an overhead of merely 8 bytes per message.

While SNEP provides pairwise authentication, it cannot provide authentication for broadcasts, because if the key is shared among several nodes, compromising any of them would allow the attacker to masquerade as any node in the group. Broadcast authentication is therefore handles using μTESLA.

Because of the nature of broadcast, asymmetry of information is very important. The sender (typically the base station or cluster head) must be able to generate a signature that the other nodes can verify, but not create on their own.

μTESLA accomplishes this using a one-way function $F(\cdot)$. It generates a sequence of keys to be used in the MAC in the following way:

$$K_n = F(K_{n+1}). \tag{1}$$

It is important that the keys are used in reverse order, because this assures that a key cannot be used to predict future keys ($F(\cdot)$ is one-way). In addition, the keys are disclosed periodically, rather than with each packet, because this saves the unnecessary overhead. An example of μTESLA can be seen in figure 1.

In figure 1 we can see that if a node can buffer P1 through P5, it only needs to hear K_4 to be able to authenticate all of the previous packets. Having each key represent a given time frame rather than a given packet assures that even if a packet is missed, all received packets can be authenticated.

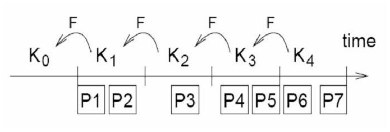

Fig. 1. An example of μTESLA [7]

B. Maintaining Service Availability

Because of the nature of attacks to service availability, most can be dealt with without specific protocols to defend against them. Bogus routing information will be avoided if the packets used to determine routing are properly authenticated. Homing can be limited through the use of encryption to conceal the contents of the packets. De-synchronization is avoided by handling the security at the MAC layer.

One of the more dangerous and hard-to-avoid attacks is physical layer jamming. The response at the physical layer would be to use frequency hopping or CDMA spread-spectrum modulations [6][9]. Forward error correction may also be included at higher layers to add redundancy, so that short bursts of noise can be compensated for [13]. However, if the attacker has unlimited power, which may be the case, this cannot be avoided, since the attacker can simply increase the power and duration of the jamming.

C. Sleep Scheduling

Sensor MAC (S-MAC)

One of the first energy-efficient MAC protocols for WSNs is S-MAC [12]. In S-MAC, there is a fixed sleep cycle (shown in figure 2 that all nodes share in which they all listen for a time (called the listen period) to initiate connection with an RTS/CTS, and then they sleep unless they are sending or receiving a packet for the rest of the time frame. This sleep time is called the sleep period. In order for this to work, there must be a loose time-synchronization between nodes. To account for the fact that the clocks may not be perfectly synchronized, the listen period must be reasonably long, and nodes do not transmit RTS's in the very beginning of the listen period.

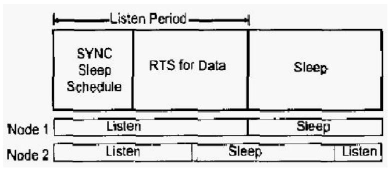

Fig. 2. An S-MAC frame [5]

The primary energy savings of this protocol come from the fact that the nodes are not always idly listening. It also happens to provide a little bit of protection from denial of sleep attacks because nodes are only vulnerable during the listen period.

Timeout MAC (T-MAC)

T-MAC [11] is another protocol that is an optimization of S-MAC. It allows nodes to go to sleep early, so they can sleep more. rather than staying awake for the entire listen period, a node can go to sleep once it has sensed an idle channel for a certain amount of time, called the adaptive timeout (TA) period. This time is defined as the longest amount of time a hidden node would have to wait before hearing the first bit of a CTS. In practice, this number is scaled up by a factor of 1.5 to handle imperfect synchronization

$$TA = 1.5 * (t_{CW\_Max} + t_{RTS} + t_{SIFS}) \tag{2}$$

An example is shown in figure 3.

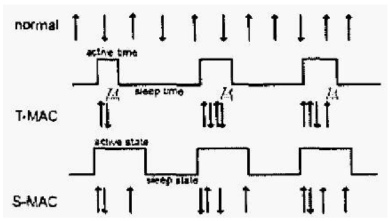

Fig. 3. T-MAC adaptive timeout example [11]

While T-MAC allows nodes to sleep more in the absence of an attacker, it is actually *more* vulnerable to attacks. This is because there is no cap on the listen time. If an attacker transmits often enough, nodes will have to stay awake forever.

Berkeley MAC (B-MAC)

An alternative approach is offered in [8]. B-MAC does not require any time synchronization. It merely sets a required sleep cycle frequency. Nodes spend most of their time in sleep mode waking up periodically to sample the channel. In order to assure a receiver will hear a transmission, the sender must transmit a longer preamble (at least as long as a single sleep cycle). An example is shown in figure 4.

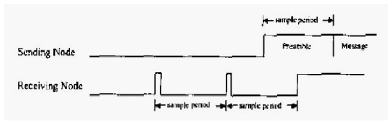

Fig. 4. An B-MAC transmission [8]

The idea behind this is that the energy spent on the longer preambles is more than made up for by the fact that nodes can sleep for most of the time they are not transmitting. While this is potentially much better than T-MAC when there is no attacker, it is equally susceptible to an attack.

Gateway MAC (G-MAC)

While previous work on energy-efficient MAC protocols had focused on energy-saving while ignoring denial of sleep attacks, G-MAC [5] presents a way to deal with these attacks. In G-MAC, the time frame is split into 2 periods: the collection period, and the distribution period(figure 5). Each cluster designates one node to be the gateway sensor(GS). This node is responsible for scheduling the transmissions for the cluster. During the collection period, nodes that want to transmit send FRTS's (future RTS's) to the cluster head, which then reserves a time slot during the distribution period for that transmission, letting the node know through a traffic indication message (GTIM) at the start of the distribution period. The GS also collects outbound traffic in this period using a regular RTS-CTS-data-ack sequence, which it forwards out of the cluster during the distribution period.

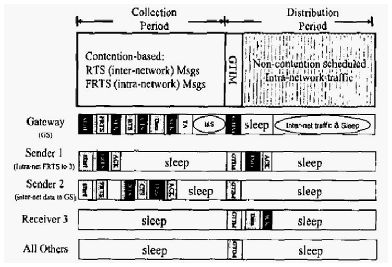
Collection Period
Distribution Period
Contention-based:
RTS (inter-network) Msgs
FRTS (intra-network) Msgs
GTIM
Non-contention scheduled Intra-network traffic
Gateway (GS)
Sender 1 (Intra-net FRTS to 3)
Sender 2 (inter-net data to GS)
Receiver 3
All Others
FRTS
RTS
CTS
ACK
TA
sleep
Inter-net traffic & Sleep

Fig. 5. An example G-MAC frame [5]

With this scheme, the non-GS nodes can all sleep during the collect period waking only to send their RTS's and outbound traffic. They then all wake up for the GTIM and stay awake only to send and receive transmissions in the distribution period. The GS has to stay awake much longer, but it can sleep during the distribution period when all outbound transmissions are done. Because this is more energy-intensive for the GS, it is necessary that the GS role rotate among all the nodes in the cluster.

This protocol performs better in defense of a denial of sleep attack, because no one can really be hurt by a bogus RTS. Nodes only stay awake to receive transmissions if they are told to by the GS in the GTIM. So potential attacks must be directed at the GS, which should authenticate the packets it receives.

IV. Cluster-Based Security

As we have seen in the G-MAC example, clusters can provide major advantages in sensor network security. In the case of G-MAC, we let the GS be the CH. The CH can also monitor the traffic coming from each MN and figure out if any of them have been compromised. It can then blacklist these nodes, isolating them from the network. In case a CH is compromised, MNs must also have the ability to decommission the CH if there are enough MNs that agree to do so [10]. This will defend against homing attacks. It is critical that several nodes agree to decommission the cluster head, because if only a few nodes are compromised, they should not be able to take down the cluster head.

When a node is removed, its transmissions will be ignored, and nothing will be sent to the node. However, it will still be able to hear and understand broadcast traffic. This can be fixed with a slight modification to μTESLA in which the function $F(\cdot)$ is not fixed at deployment time, but is rather a function that takes two arguments: a cluster key, κ and key for the broadcast K_n. We can then generate keys using

$$K_n = F(\kappa \oplus K_{n+1}). \tag{3}$$

When a node is removed, the CH can generate a new κ, and inform each of the remaining nodes individually. An analysis of this protocol modification may be included in future work.

When a node is monitoring another node, there are two ways in which it can detect misbehavior: anomaly detection and signature detection [4]. In anomaly detection, the monitoring node looks for deviations from typical behavior. This has a high probability of false alarm, so it is not often used [4]. Signature detection, on the other hand, looks for particular types of misbehavior. This leaves it susceptible to new creative types of attacks, but there are not really a lot of new different actions a misbehaving sensor node could take. The typical actions in this application would be dropping packets, duplicating packets, or causing collisions [10].

We discussed the mechanisms for intrusion detection in III, but while we know how to verify packets, we need to determine the optimal place to do the checking. It turns out that the CH is the right place to check for authenticity, as discussed in [4]. The simplest idea for intrusion detection would be to have all nodes monitor each other in promiscuous mode, meaning they listen in on all transmissions, even if they are not the recipient, but this wastes way too much energy. The other extreme would be to only check the packets at the end (base station or its neighbors), but by waiting to check the packet, we run the risk of transmitting a bad packet far too many times, consuming valuable resources. The packet should be check only a few times, and early in the path. [4] suggests that this should occur at the cluster head. All packets should be first sent to the cluster head, which checks their authenticity before forwarding them out to the rest of the network. But a malicious node can avoid this by neglecting to send its packet to the cluster head first, instead opting to send it out to another cluster. To compensate for this, every node should have a probability, p, of sending a packet to its cluster head to be checked. The higher the p, the more energy consumed in checking, and the longer the route is made, but the more likely we are to catch malicious packets. Selecting the proper p may possible be done using the method I discuss in VI.

V. A Game Theory-Based Approach

In this section I will discuss a game theory-based approach to intrusion detection presented and discussed in [2] and [3]. In this framework, intrusion-detection is looked at in the form of a 2-player non-cooperative nonzero-sum game. The two players are the intrusion detection system (IDS) of the WSN, and the attacker. The IDS wants to maintain functionality of the network by preventing attacks, while the attacker wants to disturb normal operation.

The model for the WSN is a large network of nodes sorted into clusters. When the IDS defends, it defends a cluster. Due to system limitations, the IDS can only defend one cluster at a time. The attacker can also only attack one cluster at a time.

Some key notation:

- $U(t)$: the Utility of the WSN's on-going sessions.
- C_k: the average cost of defending cluster k.
- AL_k: the average loss by losing cluster k.
- $PI(t)$: the attacker's profit for intruding
- CI: the attacker's cost to intrude
- CW: the attacker's cost to wait (opportuinity cost)

Some important assumptions to note:

- PI is $\sum_{\text{all } k} AL_k$
- $CW < PI - CI$
- $C_k \approx \gamma_k$ where γ_k is the number of previous attacks to k.

The justification for the first assumption is that the profit of the attacker should be related to the potential loss of the IDS. This metric may not be the best to use, as I will discuss in VI. The second assumption is based on the fact that if waiting were better than attacking, there would be no attacker. The final assumption is based on the idea that it is more costly to defend clusters that have been attacked before. The authors explain this by saying that the situation is analogous to insurance, but are more obvious explanation is that clusters that have been attacked have wasted some energy (due to extra transmissions or computations to defend).

The authors then go on to look at a particular cluster k in order to make a simple 2×3 payoff matrix (shown in table I).

TABLE I
Payoff matrix (for cluster k)

	Attack k	Do Nothing	Attack k''
Defend k	$U(t) - C_k$ $PI(t) - CI$	$U(t) - C_k$ CW	$U(t) - C_k - AL_{k''}$ $PI(t) - CI$
Defend k'	$U(t) - C_{k'} - AL_k$ $PI(t) - CI$	$U(t) - C_{k'}$ CW	$U(t) - C_{k'} - AL_{k''}$ $PI(t) - CI$

The rows represent the possible actions of the IDS and the columns represent the possible actions of the attacker. In each entry, the top quantity is the payoff for the IDS and the bottom quantity is the payoff of the attacker.

Because $CW < PI(t) - CI$, the attacker will always attack, so we can eliminate the middle column. His payoff is then the same no matter what. The best payoff for the IDS is when it defends the cluster that the attacker attacks. (This is rather obvious.) The result is that the IDS should always defend the "right" cluster. [2] and [3] do not offer the way to choose this cluster.

VI. The Modified Game

This game formulation is rather unsatisfying. There are a few obvious problems with it. First, the attacker benefit is independent of what the IDS does. But if the attacker's goal is to cause harm to the network, it should derive greater utility if the IDS does not defend against the attack. Secondly, the IDS should not have to defend only one cluster. If only one cluster could be defending at any given time, many extra control messages would have to be sent to coordinate the clusters. Plus, there could be a benefit to defending more than one cluster. It would just cost more resources. In the earlier discussion of cluster-based security using CHs as suggested in [4], we had assumed that all potential IDS nodes were always on. That is, all packets were checked initially at the CH.

In the modified game, each cluster will have to act independently of the others, but they are restricted to playing the strategy decided by the IDS. (Since nodes are not real people, they have no desire to put their individual utility above the rest of the network.) Each cluster is also assigned its own utility, $U_k(t)$. Each cluster also has an associated cost to defend it (i.e. energy consumption spent on defense), which we will call $C_k(t)$. We can also ignore the average loss for losing a cluster, because we can count this into the cluster utility. That is, $U_k(t)$ represents the difference in utility between having a cluster and losing it. This simplification costs us nothing as long as we assume the network is already deployed. (We never have to consider the case where a given cluster never existed.)

We will call the payoff of the IDS A and that of the attacker B.

We will also begin by looking at the one-shot game, and I will suppress the time dependence for notational brevity. We will also assume full information.

IDS payoff at cluster k:

- If we defend cluster k, we assume the attack is repelled, and our payoff is $U_k - C_k$.
- If we do not defend k and the attacker does not attack k, our payoff is U_k.
- If we do not defend k and the attacker attacks k, our payoff is 0.

As we have seen before, the attacker will always attack, so we can say his payoff is always proportional to the loss of the IDS less a constant:

$$U_{attacker} = U_k \times 1\{k \text{ not defended}\} - CI \tag{4}$$

where $1\{\}$ is an indicator function.

Theorem 1: This game has no pure NE

Proof: Suppose there were a pure NE. This means that the attacker will attack a particular node, call it k. The best response of the IDS would be to defend k and not defend and other clusters. This would make the payoff of the attacker $-CI$. But the attacker can do better by attacking a different node, j, for which its payoff will be $U_j - CI$. This contradicts the assumption that this a NE.

This game may, however, have a mixed NE. In this mixed equilibrium, we will let p_k be the probability that the IDS defends cluster k and q_k the probability that the attacker attacks cluster k. ($\sum_k q_k = 1$) because the attacker attacks only one cluster, but this is not true for the p_k's. Using this notation, we can see that the expected payoff for the attacker when he attacks cluster k is

$$E[B|\text{attack cluster } k] = (1 - p_k)U_k \tag{5}$$

If $E[B|\text{attack cluster } k] > E[B|\text{attack cluster } j]$ for some j, k, the attacker will never attack j. This means the IDS should never defend j.

Let us call $\mathcal{J}$ the set of all j s.t. $p_j = 0$ and $\mathcal{K}$ the set of all k s.t. $p_k \neq 0$.

Theorem 2: $E[B|\text{attack cluster } k] = \text{ constant } \ \forall k \in \mathcal{K}$

The proof is simply the previous result that if $E[B|\text{attack cluster } k] > E[B|\text{attack cluster } j]$, $p_j = 0$, so $j \notin \mathcal{K}$.

So the NE strategy for the IDS is to make a set of nodes, $\mathcal{K}$, s.t. $(1-p_k)U_k = X \ \ \forall k \in \mathcal{K}$ and $(1-p_k)U_k \geq U_j \ \ \forall j \notin \mathcal{K}$, for some X.

X can be a predetermined parameter (we will discuss how to select X later, and each cluster should know its own utility. The clusters can then each choose their probability $p_k = 1 - X/U_k$.

The attacker will then attack the clusters in $\mathcal{K}$ each with equal probability:

$$q_k = \frac{1}{m} \ \text{ where } m = |\mathcal{K}| \tag{6}$$

and will never attack any of the clusters not in $\mathcal{K}$:

$$q_j = 0 \ \ \forall j \notin \mathcal{K} \tag{7}$$

The resulting expected payoff for the attacker is $E[B] = X$.

The expected payoff for cluster k will be

$$\begin{aligned} E[\text{payoff of cluster } k] &= p_k(U_k - C_k) + (1 - p_k)(U_k \tfrac{m-1}{m} + 0\tfrac{1}{m}) \\ &= p_k(U_k - C_k) + (1 - p_k)(U_k \tfrac{m-1}{m}) \end{aligned} \tag{8}$$

The expected total utility of the clusters in $\mathcal{K}$ is then

$$\begin{aligned} E[\text{payoff of } \mathcal{K}] &= \textstyle\sum_{k\in\mathcal{K}}[p_k(U_k - C_k) + (1 - p_k)(U_k \tfrac{m-1}{m})] \\ &= \textstyle\sum_{k\in\mathcal{K}}[(1 - \tfrac{X}{U_k})(U_k - C_k) + \tfrac{X}{U_k}(U_k \tfrac{m-1}{m})] \\ &= \textstyle\sum_{k\in\mathcal{K}}[U_k - C_k - X + X\tfrac{C_k}{U_k} + X\tfrac{m-1}{m}] \\ &= m(X\tfrac{m-1}{m} - X) + \textstyle\sum_{k\in\mathcal{K}}[U_k - C_k + X\tfrac{C_k}{U_k}] \\ &= -X + \textstyle\sum_{k\in\mathcal{K}}[U_k - C_k + X\tfrac{C_k}{U_k}] \end{aligned} \tag{9}$$

This makes the total payoff of the IDS

$$E[A] = -X + \sum_{k\in\mathcal{K}}[U_k - C_k + X\frac{C_k}{U_k}] + \sum_{k\notin\mathcal{K}} U_k \tag{10}$$

In order to understand what this means, we will look at the expected payoff when $p_k = 1 \ \ \forall k$:

$$E[A|p_k = 1 \ \ \forall k] = \sum_{\text{all } k}[U_k - C_k] \tag{11}$$

The difference in payoff is then

$$\begin{aligned} E[A] - E[A|\text{non-random}] &= -X + \textstyle\sum_{k\in\mathcal{K}}[U_k - C_k + X\frac{C_k}{U_k}] + \sum_{k\notin\mathcal{K}} U_k - \sum_{\text{all }k}[U_k - C_k] \\ &= -X + \textstyle\sum_{k\in\mathcal{K}}[U_k - C_k + X\frac{C_k}{U_k}] - \sum_{k\in\mathcal{K}}[U_k - C_k] + \sum_{k\notin\mathcal{K}} C_k \\ &= -X + \textstyle\sum_{k\in\mathcal{K}}[X\frac{C_k}{U_k}] + \sum_{k\notin\mathcal{K}} C_k \\ &= X(\textstyle\sum_{k\in\mathcal{K}}[X\frac{C_k}{U_k}] - 1) + \sum_{k\notin\mathcal{K}} C_k \end{aligned} \tag{12}$$

There are a few observations to be made about this result. We see that this goes to $-X$ as $C_k \to 0$. This makes sense because if we are not limited by the cost of defending a cluster, we should defend all clusters.

However,the quantity is positive for larger C_k, smaller U_k and a large number of total nodes These conditions are likely to hold in sensor networks. The energy constraints make C_k high, and the fact that there are a lot of nodes makes the per-cluster utility, U_k, small for certain applications.

When this quantity is positive, it increases with X. At first this seems counter-intuitive because X is the utility of the attacker. But this makes sense because we are getting a benefit from not being over-protective. Because the game is nonzero-sum, we can do better by letting the attacker have some utility so that we do not spend too many critical resources.

If we have confidence in our security system's ability to contain an attack to within a single cluster we can now have a trade-off between securing clusters with high probability, and simply deploying more nodes. Because battery life is severely constrained and sensor nodes are relatively cheap, it may be a beneficial tradeoff to simply deploy more nodes.

We can also look at a variant of this game in which the attacker can attack more than one cluster. Let h be the number of clusters he can attack. Because attacking is always better than waiting, he will always attack exactly h unique clusters. This results in

$$q_k = \frac{h}{m} \quad \forall h \in \mathcal{K} \tag{13}$$

Doing the math out again yields a utility gain for the IDS of $X(\sum_{k\in\mathcal{K}}[X\frac{C_k}{U_k}] - h)$.

We have looked only at the one-shot game. If we assume full information, the result should be the same for the repeated game. I will leave it for future work to study the game without full information.

VII. Conclusion

We have seen that WSNs have special vulnerabilities that do not exist in wire-line networks. We cannot, therefore, simply transfer all our protocols for wire-line networks to WSNs. Protocols must be designed with low computational power and low energy requirements in mind. In this paper we have seen some of the protocols that are used, as well as some ways to determine where to check packets, including a new game theoretic approach in which we saw that by allowing the attack to have some utility, we are able to increase ours through energy saving for sufficiently large, resource constrained networks.

References

[1] Agah, Das, and Basu, "A game theory based approach for security in wirelss sensor networks"
[2] Afrand Agah, Sajal K. Das and Kalyan Basu, "A Non-cooperative Game Approach for Intrusion Detection in Sensor Networks"
[3] Afrand Agah, Kalyan Basu and Sajal K. Das, "Preventing DoS attack in Sensor Networks: A Game Theoretic Approach"
[4] Farooq Anjum, Dhanant Subhadrabandhu, Saswati Sarkar *, Rahul Shetty, "On Optimal Placement of Intrusion Detection Modules in Sensor Networks", Proceedings of the First International Conference on Broadband Networks (BROADNETS04).
[5] Michael Brownfield, "Wireless Sensor Network Denial of Sleep Attack", Proceedings of the 2005 IEEE Workshop on Information Assurance and Security United States Military Academy, West Point, NY.
[6] R. Negi and A. Perrig, "Jamming analysis of MAC protocols", Carnegie Mellon Technical Memo, 2003.
[7] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, J.D. Tygar, "SPINS: Security Protocols for Senseor Networks", Mobile Computing and Networking 2001, Rome, Italy.
[8] J. Polstre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks", ACM SENSYS 2004.
[9] M. Stahlberg, "Radio Jamming attacks against two popular mobile networks", Helsinki University of Tech. Seminar on Network Security, Fall 2000.
[10] Chien-Chung Su, Ko-Ming Chang, Yau-Hwang Kuo, "The New Intrusion Prevention and Detection Approaches for Clustering-based Sensor Networks", IEEE Communications Society / WCNC 2005.
[11] T. van Dam and K. Langedoan, "Energy-efficient MAC: An adaptive energy-efficient MAC protocol for wireless sensor networks", ACM SENSYS, November 2003.
[12] Y. Wei, J. Heiermann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks", INFOCOMM, June 2002.
[13] A. Wood and J. Stankovic, "Denial of Service in sensor networks", IEEE COmputer, October 2002
[14] Yanchao Zhang, "Location-Based Compromise-Tolerant Security Mechanisms for Wireless Sensor Networks", IEEE Journal on Selected Areas in Communications, Vol. 24, No. 2, February 2006.