Insider Threats against Trust Mechanism with Watchdog and Defending Approaches in Wireless Sensor Networks

Youngho Cho and Gang Qu
Department of Electrical and Computer Engineering
and Institute for Systems Research
University of Maryland, College Park, USA
e-mail: {youngho, gangqu}@umd.edu

Yuanming Wu
School of Optoelectronic Information
University of Electronic Science and Technology
Chengdu, Sichuan, China
e-mail: ymwu@uestc.edu.cn

***Abstract*—Trust based approaches have been widely used to counter insider attacks in wireless sensor networks because traditional cryptography-based security mechanisms such as authentication and authorization are not effective against such attacks. A trust model, which is the core component of a trust mechanism, provides a quantitative way to evaluate the trustworthiness of sensor nodes. The trust evaluation is normally conducted by watchdog nodes, which monitor and collect other sensors' behavior information. Most existing works mainly focus on the design of the trust models and how these models can be used to defend against certain insider attacks. However, these studies are empirical with the implicit assumption that the trust models are secure and reliable. In this paper, we discuss several security vulnerabilities that watchdog and trust mechanisms have, examine how inside attackers can exploit these security holes, and finally propose defending approaches that can mitigate the weaknesses of trust mechanism and watchdog.**

Keywords-inside threats; trust mechanism; sensor networks

I. INTRODUCTION

Insider threat is an important security issue in wireless sensor network (WSN) because traditional security mechanisms, such as authentication and authorization, cannot catch inside attackers who are legal members of the network. Inside attackers can disrupt the network by dropping, modifying, or misrouting data packets. This is a serious threat for many applications such as military surveillance system that monitors the battlefield and other critical infrastructures.

Trust mechanism with the notion of trust in human society has been developed to defend against insider attacks [11, 12, 15, 24]. Since WSNs consist of hundreds or thousands of tiny sensor nodes, the trust mechanism is often implemented as a distributed system where each sensor can evaluate, update, and store the trustworthiness of other nodes based on the trust model.

In general, trust mechanism works in the following three stages 1) node behavior monitoring, 2) trust measurement, and 3) insider attack detection. Watchdog [8] is a popular monitoring mechanism for the first stage. The other two stages are processed by a trust model such as beta trust model [4] and entropy trust model [10] using the data collected by the watchdogs. In such trust mechanism, if an inside attacker A keeps dropping packets from its neighbor N, watchdog in N will monitor and record this misbehavior by node A (stage 1). Then, node N will lower A's trust value (stage 2) and when the trust value goes below a trust threshold, N will consider node A untrusted and remove it from its neighbor list (stage 3).

However, there are several weaknesses in this seemingly sound mechanism. First, watchdog has some security vulnerabilities due to inherent weaknesses of WSNs such as distributed sensors, limited transceiver range, and multi-hop routing [5, 8]. Second, no trust model can completely prevent inside attackers from dropping packets. This is because for a particular packet drop, we cannot distinguish whether it is dropped by an attacker or a result from contention or noise. Thus, an inside attacker can disguise its malicious behavior behind network traffic or noise. Third, we cannot ignore the fact that insiders have internal knowledge about our network and security mechanisms against attacks. By exploiting such knowledge, inside attackers can launch their attacks intelligently to avoid being detected.

We observe that many existing trust models adopting watchdog as their monitoring mechanism do not explicitly address these weaknesses. Our goal in this paper is to demonstrate how serious insider attacks can be in WSNs even with the presence of trust mechanism and watchdog, and to introduce defending approaches to improve the trust mechanism.

II. BACKGROUNDS

A. *Packet forwarding in WSN with trust mechanism*

To understand the behaviors of inside attackers, we first explain how a sensor node forwards packets to another node in a WSN with trust mechanism. We assume that CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) is used for wireless channel access in WSNs [1].

We consider the scenario that a source node S reports its collected information (packets) to the base station (BS). The routing path from S to BS is constructed following some routing algorithm that allows multi-hop communication for energy efficiency. For each intermediate node A in the routing path, let NS be its Neighbor Set consisting of all its neighbor nodes and FS (Forwarding Set) be the set of candidate nodes that the packet may be forwarded to. Apparently, FS is a subset of NS.

To forward packets to BS, node S first chooses a node T from its FS as the next hop and sends the packet to T. Then

DOI 10.1109/SPW.2012.32

S starts monitoring T's behaviors. In the case when T receives the packet, it will send an acknowledgement message (ACK) back to S, and then forward the packet to the next node in its FS toward the BS. When the watchdog of S can overhear T's forwarding, S considers T forwarded its packet toward the BS and increases T's trust value. If S does not get the ACK message from T, S retransmits the packet up to a pre-determined number of times, three for example. If S does not get the ACK message even after the maximum retransmissions, S discards the packet and decreases T's trust value based on the trust mechanism we will elaborate later. If S gets the ACK message but does not overhear T's forwarding, S will reduce T's trust value.

B. *Insider attacks in WSNs*

To secure our network, we assume our WSN is equipped with cryptography-based authentication and authorization to defend against outside attackers launching eavesdropping or packet modification [5]. In this WSN, outside attacks may be limited to directly damaging sensors by physical strike or jamming. Meanwhile, inside attackers have some advantages compared with outside attackers. First, inside attackers damage our network stealthily since they can avoid our authentication and authorization and it is not easy to expect their attack patterns. Second, inside attackers can not only damage sensors but also disrupt our network by dropping critical packets or modifying packet information maliciously.

Inside attackers can launch various types of attacks in active (modification, packet drop, or misrouting) or passive (eavesdropping) way. While modification, misrouting, and eavesdropping can be prevented to some extent by the authentication and authorization, it is tricky to counter packet drop attacks because for a particular packet drop, we cannot distinguish whether it is dropped by an attacker or a result from collision or noise. In addition, inside packet drop attackers who are well positioned in the network (e.g., near BS) can significantly degrade network performance such as packet delivery rate due to their repeated packet drops.

There are several types of packet drop attacks such as balckhole attack, grayhole attack, and on-off attack [10, 15] as described in TABLE I. Compared to blackhole attack, it is harder to detect grayhole attack and on-off attack due to their complicated attack patterns. Moreover, packet drop attacks have evolved to intelligently drop packets by exploiting inside knowledge about our network and security mechanism to avoid being detected [15]. For this reason, in this paper, we mainly focus on inside attackers' packet drop attacks.

TABLE I. INSIDER PACKET DROP ATTACKS AND DESCRIPTION

Attack	Description
No attack	Forward all packets
Blackhole attack	Drop all packets
Grayhole attack	Drop (specific) packets randomly
On-off attack	Drop all or some portion of packets periodically

C. *Watchdog*

Marti et al [8] introduced a monitoring mechanism known as *watchdog* to identify misbehaving nodes in wireless ad hoc networks. In their approach, each sensor node has its own watchdog that monitors and records its one hop neighbors' behaviors such as packet transmission. When a sending node S sends a packet to its neighbor node T, the watchdog in S verifies whether T forwards the packet toward the BS or not by using the sensor's overhearing ability within its transceiver range.

In this mechanism, S stores all recently sent packets in its buffer, and compares each packet with the overheard packet to see whether there is a match. If yes, it means that the packet is forwarded by T and S will remove the packet from the buffer. If a packet remains in the buffer for a period longer than a pre-determined time, the watchdog considers that T fails to forward the packet and will increase its failure tally for T. If a neighbor's failure tally exceeds a certain threshold, it will be considered as a misbehaving node by S.

Watchdog works similarly with trust mechanism in that trust model evaluates each sensor's trustworthiness based on the past behaviors in much sophisticated ways. In this paper, to avoid any confusion, we consider that watchdog is a component in the trust mechanism and it is responsible for node behavior monitoring.

D. *Trust mechanism*

In general, trust mechanism works in the following stages.

1) Node behavior monitoring: Each sensor node monitors and records its neighbors' behaviors such as packet forwarding. This collected data will be used for trustworthiness evaluation in the next stage. Watchdog is a monitoring mechanism popularly used in this stage. The confidence of the trustworthiness evaluation depends on how much data a sensor collects and how reliable such data is.

2) Trust measurement: Trust model defines how to measure the trustworthiness of a sensor node. Yu et al [15] introduced several representative approaches to build the trust model, which include Bayesian approach, Entropy approach, Game-theoretic approach, and Fuzzy approach. The trust value of a node may be different when we use different trust models. For example, when a node is observed to forward the packet s times and drops the packet f times, the beta trust model [4] will assign trust value T ($0 \leq T \leq 1$) to this node using the following formula

$$T = \frac{s+1}{s+f+2}. \tag{1}$$

Meanwhile, entropy trust model [10] uses entropy function $H(p)$, whose input p is the trust value that can be obtained from beta trust model, to determine the trust value T. The entropy function $H(p) = -p\log_2 p - (1-p)\log_2(1-p)$ and the trust value T ($-1 \leq T \leq 1$) is defined by

$$T = \begin{cases} 1 - H(p), \mathit{for}\, 0.5 \leq p \leq 1; \\ H(p) - 1, \mathit{for}\, 0 \leq p < 0.5. \end{cases} \tag{2}$$

3) Inside attack detection: Based on the trust value, a sensor node determines whether its neighbor is trustworthy for collaboration (such as packet forwarding). If a neighbor's trust value is less than a certain threshold θ_T, it will be considered as an untrusted or malicious node. Depending on the WSN's trust mechanism, the detection of such insider attacker may or may not be broadcast to the rest of the nodes in the WSN.

III. VULNERABILITIES OF TRUST MECHANISM

A. Limitations of watchdog

In [8], Marti et al pointed out that watchdog has the limitation of not being able to detect a misbehaving node in the presence of the following cases. We examine each case using the path S→A→B→C→BS in Fig. 1 as an example.

1) Ambiguous collision: Consider the situation that A forwards a packet to B, and then starts to overhear whether B will forward the packet to C. However, when B forwards to C, A may not overhear this transmission if other neighbors (such as S) send packets to A at the same time. This ambiguous collision may mislead A to conclude that B is malicious, which may not be correct.

2) Receiver collision: Similar to the above case, collision may also occur at the receiver side C resulting C does not receive the packet correctly. A can only overhear that B has forwarded the packet, but A cannot tell whether C has received. When this happens, (malicious) node B can intentionally skip retransmissions or (malicious) node C can generate collision on purpose to avoid receiving the packet and to force B into retransmitting.

3) Limited transmission power: If B adjusts its transmission power such that A can overhear but C cannot receive, B can drop packets and increase its trustworthiness (to node A). In geographic routings where every node knows the positions of itself and its neighbors, B can easily launch this attack by selecting a node C from its FS such that *dist* (B, C) > *dist* (B, A) where *dist* (i, j) is a distance between node i and j.

4) False misbehavior: This case happens when a malicious node intentionally reports that other nodes are misbehaving. For example, A may report B is dropping packets although B is not. Then, A's neighbor such as node S, who cannot directly communicate (and thus monitor) B, will consider B malicious.

5) Collusion: Multiple colluding attackers can launch more sophisticated attacks. For example, two malicious colluding node A and B can completely deceive S if A forwards all packets from S to B, but B drops all the packets. Because S cannot overhear B's misbehaviors, S will not consider A and B malicious.

6) Partial dropping: Instead of dropping all packets, B can drop only some packets such that the failure tally will not exceed the detection threshold of A's watchdog. This is similar with grayhole attack.

B. Vulnerabilities in trust mechanism

We now examine the security weaknesses in each of the aforementioned three stages of a trust mechanism.

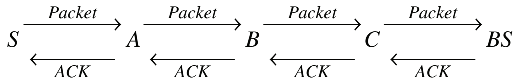

Figure 1. A routing path having three intermediate nodes A, B, and C

1) Vulnerabilities in the node behavior monitoring stage

In this stage, data will be collected for the evaluation of trust value in the later stages. Collecting reliable and trusted data will be vital. If this stage is contaminated by inside attackers, the entire trust mechanism will fail. As we focus on trust mechanisms that rely on watchdog for data collection, this stage will have the same vulnerabilities as those for watchdog in the previous section, particularly the first three: ambiguous collision, receiver collision, and limited transmission power. The other weaknesses of watchdog are closely related to the next two stages.

2) Vulnerabilities in the trust measurement stage

The key threat in this stage is that an inside attacker may figure out the trust mechanism and the associated parameters such as the trust (or trustworthiness) threshold θ_T being used. Due to the simplicity of the data collected in the previous stage and the limited available trust model (normally with low computation complexity), it will not be hard to reverse engineer to do this. To make it even worse, it is not necessary for the inside attacker to know the exact information of the trust model to launch insider attacks without being detected. For example, dropping packets as long as the insider attacker estimates its trust value will be above the estimated trust threshold. We will elaborate more about this next.

3) Vulnerabilities in the inside attacker detection stage

In this stage, a node is classified as either trustful or distrustful. The value of the trust threshold (θ_T) that is used for such classification is the single most important parameter at this stage. A low θ_T will misclassify attackers as trustful nodes and a high θ_T will cause unnecessary false alarm. θ_T must be carefully determined to maximize attacker detection rate and minimize false alarm rate.

However, if an attacker gets a reasonably good estimation on the value of θ_T, insider attacks can be launched without being detected. As shown below in Table II, if the attacker assumes $\theta_T = 0.7$, after certain number of initial successful forwarding (to build a high trust value), the attacker can drop a considerable number of packets consecutively without bringing its trustworthiness to 0.7 or below. For example, with s = 1000 previous successful forwarding, the next 428 packets can be dropped without being detected by the beta trust model, and 170 packets can be dropped if the entropy model is used.

TABLE II. BLACKHOLE ATTACKER'S MAXIMUM PACKET DROPS (PACKET DROP RATE) WITHOUT BEING DETECTED BY TRUST MODELS

Trust model	Number of previous successful forwarding (s)					
	10	100	200	300	500	1000
Beta	3 (30%)	42 (42%)	85 (42.5%)	128 (42.6%)	213 (42.6%)	428 (42.8%)
Entropy	0 (0%)	16 (16%)	33 (16.5%)	50 (16.6%)	84 (16.8%)	170 (17%)

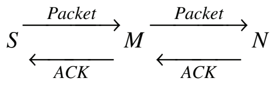

Figure 2. Single attacker

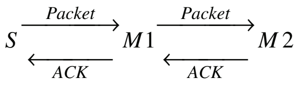

Figure 3. Two colluding attackers

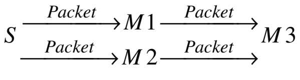

Figure 4. Three colluding attackers

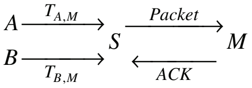

Figure 5. Reputation evaluation

IV. INSIDER THREAT ANALYSIS

In this section, we examine how inside attackers can defeat the trust mechanism combined with watchdog by exploiting the weaknesses that we have explained in section III. We first set up the goal of inside attackers, show various ways to break the network, and measure quantitatively the damage by attackers.

The goal of inside attackers is to maximize the damage on the network by preventing the collected data from reaching the BS. At the same time, they want to hide themselves so as not to be detected by the trust mechanism. As we have mentioned earlier, we focus our discussion on packet drop attacks. To measure the amount of damage caused by inside attackers, we use maximum packet drop rate (MPDR) defined below as the metric

$$MPDR\,(\%) = (N_D \,/\, N_R) \times 100 \qquad (3)$$

where N_D is the total number of dropped packets and N_R is the total number of received packets.

We now analyze three cases of inside threats: single inside attacker, multiple colluding inside attackers, and inside attacker that does more than dropping packets.

A. *Single inside attacker (M)*

Consider the malicious node M sits between the source node S and a normal node N as shown in Fig. 2. For routing protocols that use greedy forwarding approach such as the popular GPSR (Greedy Perimeter Stateless Routing) [23], node S chooses only its next hop M. S does not know to which node M will forward the packet and may not communicate directly with that node due to limited transceiver coverage.

If M receives a packet from S, sends ACK back to S, and forwards the packet to N, we consider that M behaves normally. After S sends a packet to M, there are four ways when the packet is dropped maliciously at node M: (d1) M does not receive the packet; (d2) M receives the packet but does not send ACK back to S; (d3) M receives the packet, sends ACK back to S, but drops the packet; (d4) M receives the packet, sends ACK back to S, and forwards the packet to N maliciously such that N cannot receive the packet.

The first three cases (d1–d3) will be caught by the watchdog at node S because the watchdog will not observe either M's ACK or M's forwarding behavior. When M uses these attacks, it will be caught and S will reduce M's trust value. However, M can still use these when its current trust value is high and a single drop will not lower its trust value to be below the trust threshold θ_T. In this case, MPDR depends on trust model and θ_T. We have seen this in Table II where the MPDR values are shown inside the ()'s.

In case (d4), M may be able to simultaneously avoid being detected by the watchdog and increase its trust value at node S. There are several ways to implement this. For example, M may exploit the weaknesses of watchdog we have discussed in section III (for instance, adjusts the transmission power such that S can overhear but N cannot receive the forwarded packet). M can also forward the packet to a non-existing node (or a dead node that has run out of battery) and let S overhear the forwarding. In this case, MPDR can reach 100%, which means that M can drop all the packets while it maintains high trust value from S.

B. *Multiple colluding inside attackers (M1, M2, M3, …)*

Multiple colluding inside attackers can achieve their goal much easier than a single inside attacker can. Consider two colluding attackers M1 and M2 who are neighbors in a routing path as shown in Fig. 3. M1 will ACK on the reception of packets from S and forward them to M2. This will convince the watchdog in S to increase M1's trust value. However, when the watchdog in S cannot overhear M2, M2 can use any of packet drop methods (d1–d4) and drop all the packets received from M1 without being detected by S.

However, this sequential placement of attackers may not generate the largest MPDR. In fact, the actual damage on S depends on the total number of packets that the first attacker M1 receives from S. The actual MPDR will become much lower if S distributes its packets to several neighbors to balance workloads for energy-efficiency or to avoid hidden attackers for security. For example, if we assume that S has four neighbors including one attacker M1 in FS and evenly distributes its packets to the four neighbors, M will get 25% of packets from S and the actual MPDR will be 25%.

For this reason, attackers will try to receive as many packets as possible from S in order to maximize the damage on S. In general, routing algorithm has one or more decision metrics that determine the next hop. For example, if trust is the most important decision metric, a node with the highest trust value in FS will receive all or the largest number of packets from S depending on the next hop selection algorithm. If inside attackers have this knowledge, the first attacker M1 will try to maintain a high trust value by faithfully collaborating with S and the second attacker M2 will drop S's all packets forwarded by M1. Another way to increase the damage on S is to additionally deploy inside attackers near S. Fig. 4 shows three colluding attackers M1, M2, and M3. M1 and M2 are positioned within S's one hop distance, and M3 is two hops away from S. If S has four neighbors including M1 and M2 in FS, M1 and M2 will receive at least 50% of packets from S, and M3 will drop all packets received from M1 and M2. Thus, the actual damage on S grows to more than 50%.

Meanwhile, trust mechanism in WSNs has evolved to overcome a single sensor's weakness (limited information about its neighbors) by using indirect information provided by neighbor nodes [7, 10, 15, 24]. That is, a node can get *reputation value* about its neighbor by considering both direct trust value by itself and indirect trust values (or indirect observations) by its neighbors. In Fig. 5, S has two good neighbors A and B who are also communicating with malicious node M. Then, each S, A, and B will have direct trust value about M as $T_{S,M}$, $T_{A,M}$, and $T_{B,M}$, respectively. By combining the three trust values as $R_{S,M}=f\,(T_{S,M}, T_{A,M}, T_{B,M})$ where $f\,(\cdot)$ is a reputation evaluation function, S can obtain reputation value $R_{S,M}$ about M. We show how S can utilize this information to see if M is malicious through a simple example. Consider the following situation in Fig. 5. M is smartly dropping packets received from its neighbors S, A, and B. To maximize the damage on them, M maintains it trust values at S, A, and B as slightly higher than θ_T. In this case, if S uses only its own trust value, M will not be detected. On the other hand, consider that S evaluates M's reputation by using a simple reputation evaluation function as $R_{S,M} = \alpha T_{S,M} + \beta T_{A,M} + \gamma T_{B,M} - \mathit{penalty}\ (T_{S,M}, T_{A,M}, T_{B,M})$ where $0 \leq \alpha, \beta, \gamma \leq 1$, $\alpha + \beta + \gamma = 1$, and *penalty* $(\cdot)$ is a user-defined function that grows as all inputs (trust values) are close to θ_T. In this approach, $R_{S,M}$ will be lowered below θ_T. Unfortunately, inside attackers also have evolved intelligently to avoid trust mechanism.

C. *Intelligent inside attacks against trust mechanism*

In this part, we introduce several sophisticated attacks exploiting the mathematical or logical vulnerabilities of trust mechanism. There are three types of attacks that directly disrupt trust mechanism such as bad mouthing attack, conflict behavior attack, and intelligent behavior attack [3, 7, 10, 15]. We introduce how each attack works in brief.

In *bad mouthing attack*, attackers spread negative information (trust or reputation value) about good nodes. Let us assume that A and B are malicious but M is good in Fig. 5. If A and B provide S with dishonest bad indirect trust values about M, S may falsely distrust M and discard M. Similarly, A and B may provide S with dishonest good indirect trust values about themselves to increase their trust values at S.

In *conflicting behavior attack*, attackers impair good nodes' indirect trust (or recommendation) by behaving differently to different nodes. Let us assume that an inside attacker M behaves nicely to A and behaves badly to B in Fig. 5. A and B will have different observations about M. When they exchange indirect trust values about M, A may not trust B's indirect trust value about M, and vice versa. As a result, it causes conflict opinions between two good nodes.

In *intelligent behavior attack*, attackers selectively adjust their behaviors based on intercepted or reverse engineered critical data that affect reputation evaluation, such as trust value or trust threshold. If attackers obtain the data, they can safely launch attacks without being detected by adaptively behaving nicely or badly. By definition, this attack includes most misbehaving attacks that exploit inside knowledge about the network and trust mechanism.

V. DEFENDING APPROACHES

Throughout previous sections, we have shown that even single security vulnerability in trust mechanism can be exploited by inside attackers, thus resulting in a huge damage on our network. Therefore, we must eliminate the identified vulnerabilities and have countermeasures that defend against inside attackers exploiting the security holes. In TABLE III, we list the working stages of a general trust mechanism at the first column and related security vulnerabilities and attacks at the second column. In this section, we present how we can mitigate the security vulnerabilities in each step and some existing works with their advantages and limitations (at the third column). We also introduce some research ideas.

A. *Improving stage 1: behavior monitoring*

1) *Neighbor-based monitoring*

A sender S cannot completely monitor misbehaviors of a receiver or multiple colluding attackers due to its limited overhearing distance. One way of improving this limitation is to virtually extend S's monitoring coverage by helps from other neighbors who can also monitor all forwarding participants' behaviors in a routing path. This approach can mitigate several types of colluding attacks that we have shown in previous sections. In the example of two colluding attackers M1 and M2 positioned in a routing path S→M1→M2→BS, we explained M2 can drop all packets without being detected by S due to the S's limited overhearing distance. On the contrary, in this approach, M2's misbehaviors can be detected by common good neighbors (called *guard nodes* in [21]) of M1 and M2. Moreover, the guard nodes will easily detect M1's misbehaviors, since they observe that M1 violates trust mechanism because M1 keeps forwarding packets to M2 although M2 keeps dropping all packets received from M1. For another example, guard nodes can detect whether an attacker sent a packet to a non-existing node by trying to contact the non-existing node. In addition, power-adjusting attack can be detected by guard nodes examining whether or not the strength of transmission power is enough to reach to the receiving node.

Several works [2, 14, 21] that used neighbor-based approach have been introduced in order to mitigate selective forwarding attacks. In [14], when an inside attacker drops a packet, a monitoring neighbor (called *monitor node*) alarms it to S and BS and also sends a copied packet to BS along a new routing path that is disjoint with the original routing path.

However, there are some limitations in these approaches. First, they do not address how their approaches can counter M2's selective packet drops against S. If M2 stores enough packets received from multiple nodes in its forwarding buffer, M2 can safely pinpoint S's packets by using a simple scheduling method so as not to trigger neighbor nodes' alert mechanism. To defend against M2's selective forwarding attack, neighbor nodes must be able to figure out which source node is under selective forwarding attack. In addition, a serious problem happens when neighbor nodes falsely accuse good nodes of attackers. In this case, we must have a countermeasure that not only detects selective forwarding attackers but also locates the misbehaving guard nodes.

TABLE III. SECURITY VULNERABILITIES IN TRUST MECHANISM AND DEFENDING APPROACHES

Working stages		Security vulnerabilities/Attacks	Defending approaches
Direct	1. Behavior monitoring based on watchdog	Limited overhearing/ Intentional collision, false behavior, collusion, partial dropping	Neighbor-based monitoring, acknowledgement-based monitoring, indirect observation
	2. Direct trust evaluation (or direct trust measurement)	Limited information/ Reverse engineering	Anomaly detection (e.g., consecutive failures), hiding trust evaluation mechanism (e.g., software/data obfuscation)
	3. Detection based on direct trust	Incomplete trust threshold, miss detection and false alarm/ Reverse engineering	Optimized trust threshold, dynamic trust threshold, avoidance (e.g., multipath routing), redundancy
Indirect (extended)	4. Collecting indirect information (recommendation) from neighbors	Unreliable information/ False behavior, bad mouthing, conflict behavior attack	Anomaly detection (eliminating erroneous measurement), redundancy (k fault tolerance)
	5. Reputation evaluation based on both direct and indirect information	Unreliable information/ Reverse engineering	Anomaly detection, hiding trust evaluation mechanism (e.g., software/data obfuscation)
	6. Detection based on reputation	Incomplete trust threshold, miss detection and false alarm/ Reverse engineering	Optimized trust threshold, dynamic trust threshold, avoidance (e.g., multipath routing), redundancy

2) *Acknowledgement-based monitoring*

Xio et al [13] proposed a multi-hop acknowledgement scheme to detect selective forwarding attacks. In this approach, some randomly chosen nodes (called *checkpoints*) in a routing path report ACKs back to source node S (hop by hop) by using the same but reversed routing path when they receive a packet. If a previous checkpoint does not receive ACK from a next checkpoint, it reports an alert ACK to S or BS hop by hop along the same path. Then, S figures out which nodes are malicious or suspicious based on collected ACKs from checkpoints, and then discards them. However, this approach has some weaknesses. First, while an ACK traverses back to S, insider attackers in the routing path can drop it as they dropped packets. Second, it is unclear how to accurately locate inside attackers. Third, it fails to handle when this checkpoints nodes falsely prosecute good nodes.

B. *Improving stage 2: direct trust evaluation*

1) *Anomaly detection*

To the best of our knowledge, there is no existing work focusing on defending against intelligent behavior attack. To counter this attack, first, we must prevent inside attackers from obtaining critical information such as trust value and trust evaluation procedure. However, this may not be completely achieved because insider attackers may be able to steal that information through reverse engineering; we will further explain about this later. Second, we must detect inside attackers by accurately measuring trust values of nodes and then classifying nodes into two groups (*bad* and *good*) based on the trust threshold. There must be certain unique characteristics of inside attackers since their goal must be different to that of normal nodes. Thus, a desired trust model must capture the unique aspects of inside attackers and consider them for trust evaluation.

Consecutive failures

We introduce one abnormal characteristic of packet drop attackers, *consecutive failures* (or *consecutive drops*). We believe that handling consecutive failures appropriately improves the early detection ability of a trust model because of two reasons. First, most packet drop attacks such as blackhole, grayhole, and on-off attack generate a certain degree of consecutive failures. Second, if the size of consecutive failures n grows, our belief that the node generating the n consecutive failures is not a normal node (that is, it is an attacker or a faulty node) will also grow based on the following probabilistic reasoning. Assuming that $P[f]$ is the probability that a normal node generates a failure, as n grows, the probability that the n consecutive failures happens ($P[f]^n$) decreases exponentially.

Meanwhile, we observe that two trust models (beta trust model and entropy trust model) do not address consecutive failures as (1) and (2). Consider the two observations that contain 10 successes and 10 failures: fsfsfsfsfsfsfsfsfsfs and ssssssssssffffffffff. Both models will equally treat them although the latter looks more suspicious due to the recent 10 consecutive failures according to the above reasoning. Moreover, it is often assumed that inside attackers launch attacks after they develop high trust to avoid being easily detected [24]. This assumption also supports our argument.

We show how the two trust models fail to quickly detect a naive inside packet drop attacker through a simple analysis. Suppose that a node' trust value is approximately 1 (the node is very trustful) after it successfully forwarded 1000 packets (that is, s = 1000), then the node starts dropping packets. As the number of consecutive failures n goes from 1 to 20, the upper two curves in Fig. 6 show how their trust values T drops; Trust values in beta trust model and entropy trust model are calculated by (1) and (2), respectively. Surprisingly, after 20 consecutive failures, the trust values in beta trust model and entropy trust model are 0.979 and 0.927, respectively. Even for a very noisy channel with $P[f]$ = 0.5, the event of 20 consecutive drops happens with probability 0.5^{20} ($\approx 10^{-8}$). Therefore, we need to build a new trust model that considers consecutive failures. Such model will give significant penalty on a node's trust value when consecutive failures happen as shown in the bottom curve in Fig 6.

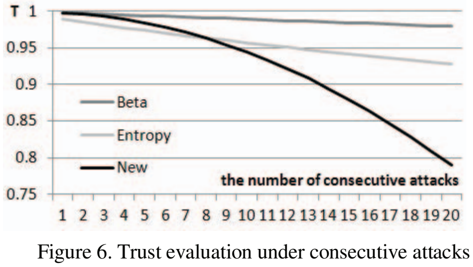

Figure 6. Trust evaluation under consecutive attacks

Like this manner, we may find inside attackers' abnormal behavioral characteristics that make them distinguishable from normal nodes. Various anomaly detection techniques [18, 19, 20] have been used to detect inside attackers in WSNs. Direct and indirect information can be used together to detect anomaly behaving nodes which are statistically deviated from normal nodes. However, most anomaly detection techniques demand nontrivial computation cost and message exchanges leading to high power consumption. Therefore, it is crucial to make them very suitable for WSNs.

2) Hiding trust evaluation function from inside attackers

If an inside attacker figures out how the trust evaluation function works, it can estimate its trust values at its neighbors based on its packet drop attack rate. Once the attacker knows its estimated trust values at others, it can intelligently adjust its attack rate and thus it will not be detected by its neighbors. Therefore, we must hide all critical functions (including source codes) appropriately from even the owner (sensor). In fact, a sensor node may not need to know the trust evaluation function or exact trust values to do certain trust-related operations. For example, for trust-based packet forwarding, a sender just needs to pick up a trustful next hop to send its packet to BS via the next hop instead of knowing the exact trust value of the next hop or how the next hop is chosen. That is, we must allow only authorized node to access only necessary information. This can be achieved by using cryptography, authentication, and authorization.

However, there remains a risk that inside attackers may reverse engineer trust evaluation function to figure out how it works and estimate its own trust value at its neighbors in order to avoid being detected. Obfuscation [17, 27] can defend against inside attackers' reverse engineering by making internal software (layout, design, and control) and data ambiguous and hard to interpret by the attackers. In addition, various software protection techniques such as watermarking, application performance degradation, and anti-debugging can be used in order to detect unauthorized access to the software, alter the software when it is accessed in unauthorized ways, and prevent attackers from using a debugger that tracks the execution of software by detecting the use of the debugger, respectively [16, 28].

C. Improving stage 3: detection based on direct trust

1) Optimized trust threshold

In stage 3, one problem is how we determine a trust threshold. In Fig. 6, assuming that we use beta trust model, if we simply set the threshold to 0.5, an inside attacker who forwarded 100 packets previously will not be detected even after 100 consecutive packet drops. Meanwhile, if we set the threshold to 0.99, there will be a high false alarm. Determining the value of threshold depends on applications that we use. For example, we may have a high threshold if the cost introduced from a high false alarm is very low in the application. Since there is a trade-off between detection and false alarm, we must make our best effort to find a trust threshold that maximizes detection rate and minimizes false alarm rate. A reasonable trust threshold can be determined theoretically or by well-designed simulation by considering our network environment and applications.

2) Static trust threshold vs. dynamic trust threshold

A trust threshold can be designed in static manner or dynamic manner. Static trust threshold might be optimal only for limited cases that we consider in the simulation. As a result, it may not be good for unconsidered situations. Meanwhile, dynamic trust threshold that adaptively changes according to situations in our network may have reasonably good results, although it may not be optimal for all situations. However, since dynamic trust threshold will be frequently computed, it must be designed in an energy-efficient way.

D. Improving stage 4 and 5:collecting indirect information from neighbors and reputation evaluation

In stage 4 and 5, for reputation evaluation, a node additionally utilizes indirect information (trust values or observations) from its neighbors. To defend against bad mouthing attack and conflict behavior attack mentioned in section IV, a couple of methods can be considered as follows.

First, we should consider only trustful indirect information provided by trustful neighbors. This is obvious because the information from distrustful neighbors will corrupt reputation evaluation. Second, indirect information can be weighted according to the trust level of the information provider [4]. Third, redundancy and statistical methods can be used for detecting those attacks. Reputation, which is obtained by both direct and indirect trust, can defend against bad mouthing attack because the attacker's misbehaviors will be different with what other neighbors observed [7]. In addition, if the number of good neighbors is larger than that of bad attackers, the bad mouthing attack can be mitigated and detected by majority voting or some statistical methods. Fourth, using multiple trust values on multiple types of behaviors is recommended in practice since a node might be distrustful for one behavior while it is trustful for another behavior [24]. For example, Sun et al [10] considered a special type of direct trust (recommendation trust) which is evaluated by nodes' past recommendation behaviors. It is calculated as $(s_r+1)/(s_r+f_r+2)$ where s_r and f_r are the number of good and bad recommendations received from the evaluated node. They showed that considering two types of trust values together better mitigates inside attacks. Finally, we introduce a general principle on how many redundancies we must have in order to defend against k colluding inside attackers disrupting our decision system. In Lamport's Byzantine agreement problem [6], $3k+1$ nodes (redundancies) are required to achieve a reliable agreement by beating k misbehaving faulty nodes by using $2k+1$ correctly behaving nodes. Thus, critical decision functions in trust mechanism must be designed based on this principle.

E. Improving stage 6: detection based on reputation

We note that this part also can be applied to the stage 3.

1) Avoidance

Regardless of how elegant detection techniques we have, inside attackers with high trust value can drop a certain portion of packets because of the weaknesses of trust mechanism that we have explained. Therefore, we must have avoidance techniques to ensure that packets eventually reach to BS. Karlof and Wagner [5] mentioned k disjoint multipath routing can completely defend against selective forwarding attacks involving at most *k* compromised nodes and still offer some probabilistic protection when there are more than *k* compromised nodes. Several works showing that multipath routing defends against inside attackers' packet dropping can be found in [22, 26]. Similarly, Sun et al [9] introduced multiple data flow scheme using multiple disjoint topologies. In this scheme, a sending node sends its packet through one or more randomly chosen topologies among the pre-established multiple topologies to mitigate selective forwarding attacks.

2) *Trade-off between redundancy and energy*

It is apparent that the more redundancies we have, the more reliable our network is. However, we must keep in mind the redundancies are the cost we must pay. For example, in *n* multipath routing, a sending node first determines *n* disjoint multiple paths from itself to BS and then sends *n* identical packets along the *n* disjoint paths. Consequently, this may introduce at least *n* times of computation complexity and power that a single path routing requires. In addition, the newly introduced workloads such as message exchanges that are required to manage disjoint paths may significantly degrade our network functions [25]. Therefore, we must utilize redundancy energy-efficiently.

VI. CONCLUSION AND FUTURE WORK

In this paper, we demonstrated how serious insider attacks can be in WSNs even with the presence of trust mechanism and watchdog, and introduced defending approaches to improve trust mechanism. We hope this paper would provide researchers who are interested in or currently working on trust mechanism with a brief, big picture about how we should improve trust mechanism to defend against inside attackers dropping packets. In the near future, we will design a reliable, energy-efficient trust mechanism for WSNs by considering the identified vulnerabilities and defending approaches in TABLE III.

VII. ACKNOWLEDGEMENT

This material is based upon work supported in part by the Air Force Office of Scientific Research (AFOSR/RSL) under Award No. #FA95501010140.

REFERENCES

[1] Azahdeh Faridi et al, "Comprehensive Evaluation of the IEEE 802.15.4 MAC Layer Performance With Retransmissions," IEEE Transactions on Vehicular Technology, Vol.59, No.8, October 2010, pp. 3917-3932.

[2] Tran Hoang Hai and Eui-Nam Huh, "Detecting Selective Forwarding Attacks in Wireless Sensor Networks Using Two-hops Neighbor Knowledge," Seventh International Symposium on Network Computing and Applications (NCA '08), July 2008, pp. 325-331.

[3] K. Hoffman, D. Zage, and C. Nita-Rotaru, "A survey of Attack and Defense Techniques for Reputation Systems," ACM Computing Surveys, Vol 41, Issue 4, 2009.

[4] A. Josang and R. Ismail, "The Beta Reputation System," In Proc. of the 15th Bled Electronic Commerce Conference, June 2002.

[5] Chris Karlof and David Wagner, "Secure routing in wireless sensor networks: attacks and countermeasures," Ad Hoc Networks Journal, Vol.1, Issue 2-3, 2003, pp. 293-315.

[6] Leslie Lamport, Robert Shostak, and Marshall Pease, "The Byzantine Generals Problem," ACM Transactions on Programming Languages and Systems, Vol. 4, No. 3, July 1982, pp. 382-401.

[7] Jie Li, Ruidong Li, and Jien Kato, "Future Trust Management Framework for Mobile Ad Hoc Networks," IEEE Communication Magazine, Vol 46, Issue 4, 2008, pp.108-114.

[8] Sergio Marti, T.J. Giuli, Kevin Lai, and Mary Baker, "Mitigating Routing Misbehavior in Mobile and Ad Hoc Networks," In Proc. of International Conference on Mobile Computing and Networking (Mobicom), 2000, pp. 255-265.

[9] Hung-Min Sun, Chien-Ming Che, and Ying-Chu Hsiao, "An efficient countermeasure to the selective forwarding attack in wireless sensor network," IEEE Region 10 Conference (TENCON), 2007, pp.1-4.

[10] Yan (Lindsay) Sun, Zhu Han, and K. J. Ray Liu, "Defense of Trust Management Vulnerabilities in Distributed Networks," IEEE Communications Magazine, Vol 46, Issue 2, 2008, pp.112-119.

[11] Denis Trček, "Trust management in the pervasive computing era," IEEE Security & Privacy, Vol. 9, No. 4, July 2011, pp. 52-55.

[12] Vijay Varadharajan, "A Note on Trust-Enhanced Security", IEEE Security & Privacy, Vol. 7, Issue 3, May/June 2009, pp. 57-59.

[13] Bin Xiao, Bo Yu, Chuanshan Gao, "CHEMAS: Identify suspect nodes in selective forwarding attacks," Journal of Parallel and Distributed Computing, 67, 2007, pp. 1218-1230.

[14] Wang Xin-sheng, Zhan Yong-zhao, Xiong Shu-ming, and Wang Liang-min,"Lightweight defense scheme against selective forwarding attacks in wireless sensor networks," Intl. Conf. on Cyber-Enabled Distributed and Knowledge Discovery (CyberC), 2009, pp. 226-232.

[15] Yanli Yu, Keqiu Li, Wanlei Zhou, and Ping Li, "Trust mechanisms in wireless sensor networks: attack analysis and countermeasures," Journal of Network and Computer Applications, Elsevier, 2011, in press.

[16] Martin R. Stytz and James A. Whitaker, "Software Protection: Security's Last Stand?" IEEE Security & Privacy, Vol 1, Issue 1, 2003, pp. 95-98.

[17] David E. Bakken, Rupa Parameswaran, Douglas M. Blough, Ty J. Palmer, and Andy A. Franz, "Data Obfuscation: Anonymity and Desensitization of Usable Data Sets," IEEE Security & Privacy, Vol 2, Issue 6, 2004, pp. 34-41.

[18] Fang Liu, Xiuzhen Cheng, and Dechang Chen, "Insider Attacker Detection in Wireless Sensor Networks," IEEE International Conf. on Computer Communications (INFOCOM), May 2007, pp. 1937-1945.

[19] Miao Xie, Song Han, Biming Tian, and Sazia Parvin, "Anomaly detection in wireless sensor networks: A survey," Journal of Network and Computer Applications 34, 2011, pp. 1302-1325.

[20] Sutharshan Rajasegarar, Christopher Leckie, and Marimuhu Palaniswami, "Anomaly Detection In Wireless Sensor Networks," IEEE Wireless Communications, 2008, pp. 34-40.

[21] Issa Khalil, Saurabh Bagchi, Cristina N. Rotaru, Ness B. Shroff, "UnMask: Utilizing neighbor monitoring for attack mitigation in multihop wireless sensor networks," Ad Hoc Networks, in press, 2009.

[22] Y. Challal, A. Ouadjaout, N. Lasla, M. Bagaa, A. Hadjidj, "Secure and efficient disjoint multipath construction for fault tolerant routing in wireless sensor networks," Journal of Network and Computer Applications 34, 2011, pp. 1380-1397.

[23] Brad Karp and H.T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," In Proc. of International Conference on Mobile Computing and Networking (Mobicom), 2000, pp. 243-254.

[24] Javier Lopez, Rodrigo Roman, Isaac Agudo, and Carmen Fernandez-Gago, "Trust management systems for wireless sensor networks: Best practices," Computer Communications, Vol 33, 2010, pp.1086 – 1093.

[25] David R. Raymond and Scott F. Midkiff, "Denial-of-Service in Wireless Sensor Networks: Attacks and Defenses," Pervasive computing, 2008, pp. 74-80.

[26] Suk-Bok Lee and Yoon-Hwa Choi, "A Resilient Packet-Forwarding Scheme against Maliciously Packet-Dropping Nodes in Sensor Networks," In Proc. of the fourth ACM workshop on Security of ad hoc and sensor networks (SANS), 2006, pp. 59-69.

[27] Vivek Balachandran and Sabu Emmanuel, "Software Code Obfuscation by Hiding Control Flow Information in Stack," IEEE Intl. Workshiop on Information Forensics and Security (WIFS), 2011, pp. 1-6.

[28] Tang Jiutao and Lin Guoyuan, "Research of Software Protection," Intl. Conf. on Educational and network Technology (ICENT), 2010, pp. 410-413.