

# Predictability Trust for Wireless Sensor Networks to Provide a Defense Against On/off attack

Younghun Chae<sup>1</sup>, Lisa Cingiser DiPippo<sup>1</sup>, Yan Lindsay Sun<sup>2</sup>

<sup>1</sup>Department of Computer Science and Statistics, <sup>2</sup>Department of Electrical, Computer and Biomedical Engineering  
University of Rhode Island, Kingston, RI 02881, USA  
Email: iplusu@my.uri.edu, lcd@cs.uri.edu, yansun@ele.uri.edu

**Abstract**—In Wireless Sensor Networks (WSNs), sensors collaborate to perform various tasks, such as routing. A trust-based routing scheme can be used to route around compromised nodes that attempt to upset this collaboration. However, because wireless connections can be unreliable, the routing protocols must provide a redemption scheme to allow nodes to recover trust. Otherwise, occasional bad behaviors due to unreliable communications can result in false alarms in malicious node detection. Existing redemption schemes fail to discriminate between temporary errors and an On/off attack in which the attacker cleverly behaves well and badly alternatively. In this paper, we present a new trust redemption scheme that cannot only discriminate between temporary errors and an On/off attack, but can also provide a flexible design.

**Keywords**—wireless sensor network; secure routing; trust; redemption; On/off attack

## I. INTRODUCTION

Wireless Sensor Networks (WSNs) are used in applications where environmental data is sensed, shared, fused, and routed to other nodes in the network. Because of the distributed nature of these networks, they require collaboration on many levels in order to process the data. Nodes collaborate in tasks such as data fusion, algorithm computation, target detection, and target tracking. Further, because of the limited resources of the nodes in a WSN, data cannot always be sent from a source node to a destination node directly. Rather, the data must be routed through multiple hops, requiring all nodes in the path to collaborate in the goal of getting data to the intended destination.

In many WSN applications, such as military surveillance, there may be adversaries that wish to attack the routing of data through the network in order to prevent it from getting to its destination. The possible attacks on WSN routing have been classified into two categories by F. Hu and N. Sharma [1]. The first category is external attacks, which involve an attack node that is injecting or extracting information. The second category is internal attacks, which involve an attack node that normally participates in the network but behaves abnormally. This type of internal attack can be more nefarious because it can be harder to detect, and because it can adversely affect the processing of the data that is being routed. For example, in military surveillance WSN, it is crucial for data to be delivered correctly and timely so that the collaborating nodes can use accurate data in the target detection and tracking algorithms.

There has been much research towards defenses against both types of attack. Cryptography [2-5] is a possible solution

for defending against external attacks. However, this requires expensive calculation and resource costs for sensor nodes [6-8], and it cannot provide defense against internal attacks when the secure keys are exposed. Thus, Secure Routing Protocols (SRPs) [9-15] are employed for defending against internal attacks. Most, but not all, SRPs are based on the reputations of neighboring nodes, computed by observing behaviors of the neighbors. A source node finds a trustworthy route to the destination node by using this information. When nodes want to collaborate in a WSN, or any type of network, trust is a very important characteristic – node A will be more likely to collaborate with node B than node C if node A trusts B more than C.

When these two secure schemes are employed by a WSN, the length of the transaction path becomes a critical consideration. Since each node needs to spend calculation energy and time for cryptography, the length of the path will affect the network lifetime and the transaction time. Thus, an SRP should try to find and maintain the shortest transaction path while at the same time providing a defense against internal attacks.

When a SRP is based on the trust of neighboring nodes, the choice of routing path should be dynamic, because a wireless network can be unstable. One or a few occasional bad behaviors of a neighbor node will lead to reduction in trust value, but should not make this node excluded from the routing path forever, especially if it is on the shortest path to the destination. Therefore, a SRP needs to employ a trust redemption scheme [10-13] to allow faulty nodes to recover trust under certain conditions.

Of course, a smart attacker could attempt to trick the trust redemption scheme by alternating good and bad behaviors so that trust is always redeemed just before another attack occurs. This type of attack is referred to as an On/off attack. Most current trust redemption solutions fail to effectively discriminate between an On/off attack and temporary errors, especially when the majority of the attacker's behavior is good. In this paper, we introduce a new trust evaluation and management scheme that we call *Predictability Trust*. Our goals of this work are twofold. The first goal is that a node with a non-perfect trust score should have opportunities to be considered trustworthy again. The second goal is to detect an On/off attack by employing Predictability Trust.

In this paper, we present our notion of Predictability Trust and demonstrate its effectiveness against On/off attacks of various degrees. While Predictability Trust is a general solution for detecting On/off attacks, for the sake of this paper, we have

focused on defending the combination of On/off with Selective forwarding. Selective Forwarding [7] is a routing attack in which a malicious node intentionally drops a certain percentage of the packets that are intended to pass through itself. There exist advanced versions of Selective Forwarding, such as Black hole and Sinkhole [7, 16]. In this paper, however, we consider only the simple version of Selective Forwarding because it is easy to be detected by using direct observation [15]. When combined with the On/off attack [9] Selective Forwarding disrupts the trust evaluation process by behaving well (i.e. honestly forwarding packets) and badly (i.e. dropping certain percentage of packets) alternatively. This may provide more opportunities for the malicious nodes to attack a node. A malicious node performs several good behaviors, and the system believes the node is honest. Then the node attacks with another Selective Forwarding attack. After a bad behavior, the node recovers its trust with several good behaviors, and then it attacks again. The system may consider these bad behaviors to be a result of unreliable wireless channel conditions, as long as the percentage of bad behavior is not high. However, when the damages accumulated, even 10% bad behavior can hurt system performance, not to mention that multiple malicious nodes can attack at different times and each only attacks a small fraction of time. This could be fatal on a real-time surveillance system. Our protocol, Predictability Trust, is specially designed to detect an On/off attack.

This paper is organized as follow. In Section II, we discuss the redemption schemes that have been employed by previous proposals. Then, we present our protocol, Predictability Trust, in Section III and provide the simulation results in Section IV. Finally, we conclude and suggest future works in Section V.

## II. PREVIOUS RESEARCH

In this Section, we discuss the redemption schemes that have been employed by previous work. The redemption schemes can be classified in two ways. First, Behavior Based Redemption (BBR) recovers the trust based on the future behaviors. Second, Time Based Redemption (TBR) recovers the trust periodically.

### A. Behavior Based Redemption (BBR)

To understand Behavior Based Redemption (BBR), assume that a friend had a bad behavior in the past, but since then the friend has behaved very well several times. Thus, we can expect that the friend will behave well in the next behavior. Similarly in a WSN, if a node behaves very well now, we can expect the node will behave well in the next behavior, even if the node had a bad behavior in the past. A representative scheme is as follows.

- *CORE*: P. Michiardi and R. Molva [12] have proposed “a **CO**llaborative **RE**putation mechanism protocol”. CORE evaluates neighboring nodes based on direct observation, indirect observation that considers only positive reports by others, and task-specific behavior. These are compiled by a weighted trust technique, and the compiled result is used for discriminating and isolating a malicious node from the network. By assigning higher weight to the past behaviors than the recent behavior, it

makes the recent bad behavior minimize the influence on the evaluation.

Note that BBR is not a very good choice of redemption technique to use for WSN routing because it relies on subsequent behaviors to allow trust to be redeemed. In a WSN routing scheme that uses trust to make routing decisions, an untrusted node will not be used for forwarding, and therefore no new behaviors can be observed. We discuss this type of redemption scheme because it is a widely used technique for trust redemption in general.

### B. Time Based Redemption (TBR)

If we assume that a friend had a bad behavior in the past, we might decide not to trust the friend for a while. After time has passed, we expect the bad behavior was a mistake, and we give another opportunity to the friend. Time Based Redemption (TBR) can be compared to this example. We provide some time to a node to recover from a temporary error, and we give another opportunity to behave well. A representative scheme is as follows.

- *OCEAN*: S. Bansal and M. Baker [13] have proposed “**O**bservation–based **C**ooperation **E**nforcement in **A**d hoc **N**etworks”. OCEAN relies on first-hand observations, and negative behavior decreases the rating of the node larger than positive behavior increments. When the rating is below the threshold, the node is added to the faulty list, and the faulty list is broadcasted. Other nodes use this faulty list to avoid the malicious nodes. OCEAN removes a node from the faulty list by using a timeout based approach.

### C. Combined Redemption

BBR and TBR can also be combined as shown in the following scheme.

- *CONFIDANT*: S. Buchegger and J. Y. Le Boudec [10, 11] have proposed “**C**ooperation **O**f **N**odes, **F**airness **I**n **D**ynamic **A**d-hoc **N**eTworks”, which employs both BBR and TBR for trust redemption. By adapting a Bayesian reputation and trust system, CONFIDANT allows second-hand evaluation. It uses both direct observation of a node and the second-hand information from other nodes by using a weighted trust technique. This helps to evaluate a neighboring node, and the reputation is used to isolate the malicious nodes from its network. CONFIDANT allows BBR by using weighted trust evaluation. Also it allows TBR by using a fading technique that discounts all ratings periodically and upon observation by exponential decay.

The previous research described here was not designed to detect an On/off attack, and so are not able to effectively defend against the attack while at the same time allowing redemption. Our protocol uses information about past behaviors to determine how quickly to redeem trust in a neighbor node. Thus, it provides an effective and flexible defense against an On/off attack while at the same time allowing redemption.

## III. PREDICTABILITY TRUST

As we explained in Section II, previous work in trust redemption does not provide an effective defense against the On/off attack. In this section, we describe our notion of Predi-

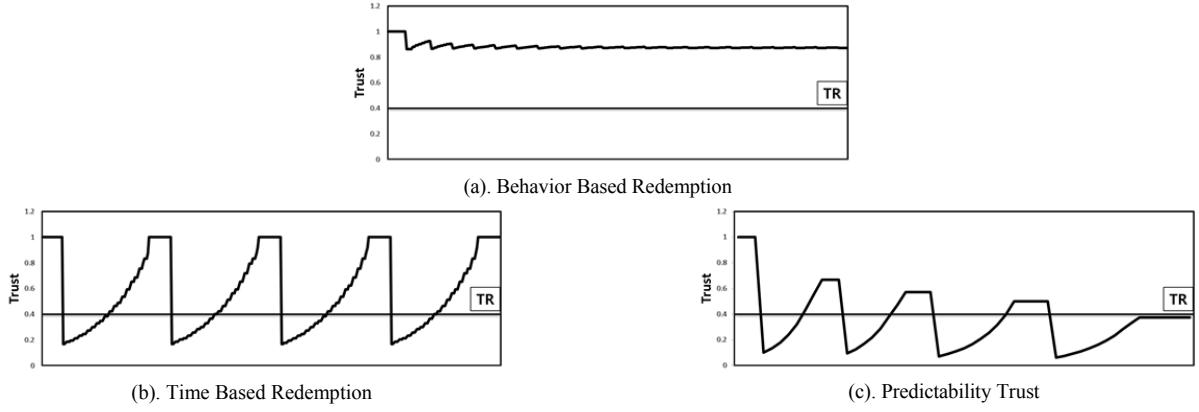


Figure 1. Examples of trust changes in On/off attack.

ctability Trust, which uses both time-based and behavior-based redemption to detect such an attack. Fig. 1 illustrates how Predictability Trust works in comparison to TBR and BBR techniques. The graphs show the overall trust in a neighbor node that is performing an On/off attack – each dip in the graph represents a bad behavior, and thus a reduction in trust. In response to this type of attack, Predictability Trust provides two important features: 1) The Overall Trust is lowered with each individual attack and 2) The amount of time it takes to redeem trust increases with each attack. In contrast to the TBR scheme, which allows full trust redemption after each attack, and the BBR scheme which does not lower trust enough with each bad behavior, Predictability Trust is able to reduce the overall trust to below an acceptable threshold level so that the neighbor node is quickly classified as malicious and not used again in the routing scheme.

In this Section, we first briefly introduce a trust-based routing protocol, called Secure Adaptive Routing Protocol (SARP) [17], which will be used as the platform to demonstrate the proposed trust. Then, we present our *Predictability Trust* (PT) protocol. While SARP provides defenses against multiple attacks by using its trust framework, PT extends the basic trust concepts used in SARP and largely improves SARP with a more robust secure scheme for detecting the On/off attack.

#### A. Secure Adaptive Routing Protocol (SARP)

SARP is a multi-dimensional trust evaluation mechanism that can adapt to dynamic changes in the trust values of nodes in the network to route data from a Data Source to a Base-Station. SARP helps a node to choose the most trusted node among its neighboring nodes to route data through towards the Base-Station. The mechanism can be applicable to various routing schemes.

1) *Trust Framework*: SARP uses a Trust Framework that manages the multi-dimensional trust. By using this mechanism, the network recognizes malicious nodes and chooses the most trusted route to the Base-Station.

a) *Trust*: SARP uses multi-dimensional trust metrics, that is, it measures various types of trust to make routing decisions. Each node computes these trust values for each of its neighbors. First, *Forwarding Trust* indicates the percentage of packets that a neighbor node has forwarded on to the next

node by monitoring the neighbor’s behavior using trust reporting. Second, *Reporting Trust* indicates how honestly a neighboring node sends reporting messages upon request. Third, *Availability Trust* indicates how a neighboring node responds to request/control messages. Fourth, *Loop Trust* indicates whether a neighboring node creates a looped route or not. Finally, the *Overall Trust* is a compilation of all of the other trust values to evaluate if a neighboring node is malicious or not. In SARP, each trust value is calculated by (1) that is based on Bayesian formulation, specifically a beta reputation system [18].

$$T_i^k = \frac{GB_i^k + 1}{GB_i^k + BB_i^k + 2} \quad (1)$$

where the  $T_i^k$  represents  $k$  type of trust of node  $i$ , the  $GB_i^k$  represents the number of good behaviors of node  $i$  that were monitored for  $T^k$ , the  $BB_i^k$  represents the number of bad behaviors of node  $i$  that were monitored for  $T^k$ . In [17], we have developed methods to determine whether a node performs a good (or bad) behavior in terms of forwarding packet (i.e. Forwarding Trust), honestly reporting the number of packets it forwarded (i.e. Reporting Trust), responding to request/control messages (i.e. Availability Trust), and not creating routing loops (i.e. Loop Trust). Since the focus on this paper is not SARP, but the new concept of Predictability Trust, we will not include a detailed introduction of these methods. It is important to point out that Predictability Trust is not limited to SARP framework, but can be applied to many trust evaluation methods.

The *Overall Trust* is calculated by (2).

$$OT = \sum_{k=1}^n T_i^k \cdot W^k, \quad \sum_{k=1}^n W^k = 1 \quad (2)$$

where the  $W^k$  represents the weight of that type of trust in the Overall Trust computation. The sum of  $W^k$  should be always 1, and SARP is able to provide a focused defense for a specific kind of attack by assigning different weights to a specific trust types.

b) *Dynamic Redemption*: SARP uses a trust redemption

mechanism known as Time Based Dynamic Redemption (TBDR), which is a variant of BBR and TBR that were discussed in Section II. TBDR varies the redemption speed depending on the current trust. A redemption factor that is based on the current *Overall Trust* is used to determine the speed of redemption. So TBDR provides a defense against an On/off attack by slowing down the redemption speed of an On/off attack node and minimizing its usage in the network. For example, when the trust value of a node is low, TBDR recovers its trust value slower than other nodes that have higher trust value. Thus, the malicious node cannot recover its trust easily after behaving badly several times, and this will reduce the usage of the malicious node in the overall transactions. While TBDR is able to slow down the On/off attack node, it is not able to detect the attack and stop using the node altogether because overall trust will eventually recover to an acceptable level.

2) *Protocol Overview*: SARP securely delivers data from a Data Source (DS) to a Base-Station (BS) through Forwarders (FDs). It does this by recognizing the shortest path to the base station (Node Discovery), creating transactions to transmit the data, and reporting on behaviors in order to compute trust values.

a) *Node Discovery*: Once the sensor nodes are deployed in the field of interest, they start the discovery phase in which each node learns which nodes are its neighbors, and how far it is from the BS. This discovery process begins when the BS broadcasts a discovery type message that includes a “0”. Any node that hears this message is considered a one-hop node and then broadcasts a discovery type message that has a “1”. A node that hears the “1” is a two-hop node, and the process continues until each node knows its shortest path to the base station.

b) *Transactions*: A transaction is the unit of data transmission and of trust reporting. When a node has a stream of data to report to a BS, the node is considered a DS, and it starts transmitting a specific number of packets. If the DS is not able to reach the BS directly, the node transmits the data to a transaction parent node. This node is called a FD, and this FD also finds a transaction parent node until the data reaches the BS. There are two criteria for choosing a transaction parent node. First, the node should be closer to the BS than the FD. The distance of each node is recognized in the *Node Discovery*, and the information is used to find the closer nodes. Second, the node should be trusted. The Trust Framework is used for finding the most trusted node among the neighbors of the DS or FDs. When there is no trusted node among the nodes closest to the BS, that is, none of the candidate parent nodes is above a predetermined trust threshold, the DS or FDs can choose a peer or child node as its transaction parent to detour around any untrusted nodes.

c) *Trust Reporting*: SARP relies on updating the trust of nodes based on performance of the nodes in transactions. When a transaction is completed that means a stream of data has been delivered from a DS to a BS. The DS begins a trust reporting process in two steps. *Requesting Trust Reports* (RTR) and *Processing Trust Reporting* (PTR). First, RTR begins

when a transaction finishes. The DS or the FD sends this requesting message to its transaction parent node, and the transaction parent node sends back its one-hop report with its own report to the requester. As a result of this process, the requester will have both one-hop and two-hop reports, when it receives the full report from its transaction parent node. Second PTR begins when the requester receives the report from its transaction parent node. The requester, which is the DS or the FD, calculates the trust value of its one-hop and two-hop neighbors depending on the report (see [17] for details of this reporting process).

### B. Predictability Trust

As mentioned earlier, SARP uses TBDR for trust redemption, and so a malicious node that combines an On/off attack with any other type of attack may remain available in the network. SARP will recover the trust of the On/off attack node because it is not able to incorporate past behaviors into the computation of trust. Thus, there is a need for detecting these sporadic malicious behaviors to prevent further attacks. To provide better security with SARP, we developed Predictability Trust (PT) with Dynamic Sliding Windows (DSW). PT is a new concept of the Trust management mechanism. It can help to predict the next behavior of a node depending on the recent behaviors, and thus detect an On/off attack and remove a malicious node from the network. Further, PT can provide flexible system design through the use of Dynamic Sliding Windows.

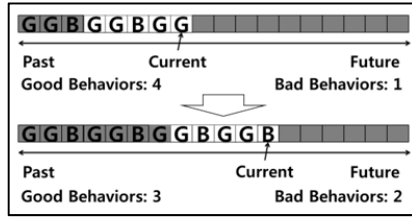
1) *Predictability Trust Calculation*: PT is computed based on how well a node’s behavior meets expectations. For example, a node’s current forwarding trust is 0.9, meaning that we (or the system) predict this node forwarding at least 90 % of the packets that are sent through it. In the next round, if this node forwards above 90% of packets, this node meets the prediction, and is considered as conducting a good predicted behavior (GPB). If this node forwards below (90- $\Delta$ )% packets, this node does not meet the expectation, and is considered as conducting a bad predicted behavior (BPB). We will count the number of GPBs conducted by node  $i$ , denoted by  $GPB_i$ , and the number of BPBs conducted by node  $i$ , denoted by  $BPB_i$ . The Predictability trust of node  $i$  is computed as in (3).

$$PT_i = \frac{GPB_i + 1}{GPB_i + BPB_i + 2} \quad (3)$$

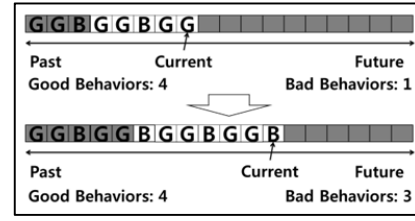
From (3), we can see that the PT of node  $i$  describes whether the current trust can accurately predict the node’s future behavior and whether a node’s behavior is consistent with his past behavior. A low PT indicates that (a) the current overall trust value is less “trusted” and therefore should be lowered, and (b) this node’s behavior is not consistent and therefore should be suspected for On/off attacks which requires an adjustment in the trust redemption.

Based on the above discussion, PT is used in two ways in our protocol: for setting the maximum value of the overall trust, and for controlling the speed of trust redemption.

2) *Overall Trust Adjustment*: The PT sets the maximum overall trust value. Overall Trust is calculated by (4).



(a). An example that how Fixed Sliding Window is changed.



(b). An example that how Fixed Sliding Window is changed.

Figure 2. Sliding windows

$$OT_i = OT_i \times PT_i \quad (4)$$

where the OT represents the Overall Trust that is computed by combining the various types of trust (recall (2)). Since the ranges of OT and PT are between 0 and 1, the OT can be at best 1.0 by (4). This allows us to use a node's predicatbility as a factor in the overall trust computation. A system designer can set a certain threshold for OT that excludes a node from being used in routing if it has a trust value below the threshold. If PT is low enough, our mechanism can lower the OT so that it is below the threshold. Thus PT has an effect on OT that is independent of the specific trust behaviors. Equations (3) and (4) are computed after every transaction in which a node is used, thus employing BBR.

3) *Redemption Speed Adjustment:* The other way in which PT is used allows it to control the redemption speed by computing a Redemption Factor (RF) of node  $i$ .  $RF$  is calculated by (5) and is used by (6) to allow trust to be redeemed at the calculated rate.

$$RF_i = (OT_i \times PT_i) \times \alpha + 1.0 \quad (5)$$

$$T_i^k = T_i^k \times RF_i \quad (6)$$

In these equations,  $0 < \alpha \leq 1$ , and it represents a mechanism to allow a system designer to control the tolerance of a system. If a system needs to be strictly secured,  $\alpha$  would be smaller than in a more tolerant system. When the  $PT_i$  is low because a node has behaved unpredictably, the redemption will take more time than for a more predictable node. By maintaining this  $PT_i$  value, the system can automatically and dynamically control the redemption speed. Equations (5) and (6) are computed periodically, thus employing time-based redemption for each trust type.

4) *On/off attack Detection:* Using the technique described in Sections III.B.2 and III.B.3, we can reduce the effectiveness of On/off attacks. In particular, the technique in Section III.B.2 reduces the Overall Trust of the On/off attackers, and the III.B.3 reduces the redemption speed of the On/off attackers. However, if we only depend on the above two techniques, the On/off attackers will not be detected in a short time. Since the trust of the attacker is reduced, the attacker will have less opportunity to be on the route and fewer observations will be made. It will take a long time to collect sufficient evidence to mark a node as malicious node. There

are two ways to address this problem. The first way is to give more opportunities to low-trust nodes to act. This is why we allow trust redemption. The second way is to adjust the method of evidence collection based on the PT, using sliding windows, as described in the next subsection. The primary idea is investigating on a longer history of bad behaviors when the PT is low.

5) *Sliding Windows:* A Sliding Window (SW) is a tool for keeping track of the past behaviors of a neighbor node. It would be best if we could observe the entire history of each neighbor, but this is unattainable because of the limited storage and processor speed of a WSN. For these reasons, we developed the SW to allow a certain number of behaviors to be stored for calculating trust. A SW updates and stores the latest behavior history. When an event is observed, the SW removes the oldest behavior in its memory and stores the latest behavior. We use two types of SW in our trust computation: a Fixed Sliding Window (FSW) and a Dynamic Sliding Window (DSW). The FSW has a fixed size observing window and is used to keep track of the good behaviors. The DSW changes size depending on the current trust, so as trust decreases the size of DSW increases. We use the DSW to keep track of bad behaviors. We are more interested in the bad behaviors than the good behaviors because they are harmful to the system, and the primary purpose of PT is the isolating of malicious nodes. However, to avoid erroneously labeling nodes as malicious, we need to be cautious in discriminating the malicious nodes. For these reasons, we developed DSW that allows us to observe more previous bad behaviors depending on the current trust value.

The FSW and the DSW keep track of both good behaviors and bad behaviors. When a new behavior is observed, it pushes out the oldest behavior from the SW. This will increase the PT if the new behavior is a good behavior and the old behavior is a bad behavior. Thus, if a neighboring node has several bad behaviors, and its trust still stays above the threshold, the PT and Overall Trust can be recovered after several good behaviors. Moreover, this ensures that the system can manage trust as the fresh information comes in.

a) *Fixed Sliding Window:* The purpose of the FSW is to count the good behaviors among the most recent behaviors. It stores both good behaviors and bad behaviors, but counts only the good behaviors. This makes it possible to consider only the fresh good behaviors, while keeping in mind the overall pattern of behaviors in the recent past. A system designer is able to provide a specific number of opportunities for bad

nodes by setting the size of FSW. For example, when the size of FSW is set to 5, the system allows at least three bad behaviors and at best seven bad behaviors.

Fig. 2.(a) shows how the FSW works. The top part of the figure shows the window status with a size of five with four good behaviors and one bad behavior. The bottom part of the figure shows the same window after four more behaviors. In this figure, the FSW is represented by the white boxes, so it forgets any behaviors in the shaded boxes. The boxes on the left of the “Current” arrow are the past behaviors that it remembers. The box that the arrow points out is the latest behavior, and the boxes on the right side the arrow represent space for future behaviors that have not occurred yet. In this example, the number of good behaviors changed from four to three after four behaviors in two-good-one-bad On/off attack.

b) *Dynamic Sliding Window*: The DSW stores good behaviors and bad behaviors, but counts only the bad behaviors. The size of the window changes dynamically as the trust of the node changes, and a system designer is able to set a maximum window size for the DSW. We have performed a set of analytical tests that indicate the tolerance to attack for a given maximum window size [19]. Thus, a system designer can choose a larger window size in order to allow for less tolerance to attack. For example, we found in our analysis that when the maximum size of the DSW is set to 65, the system does not allow a six-good-one-bad On/off attack, but it does allow an attack with a higher On/off ratio. The higher the On/off ratio, the lower the damage that can be incurred to the system. Thus, it is up to the system designer to determine how much damage the system can tolerate, and set the DSW maximum size accordingly.

$$Size = \beta \times OT + \gamma \quad (7)$$

$$MaxSize = \beta \times 1.0 + \gamma, \quad MinSize = \beta \times 0.1 + \gamma \quad (8)$$

While the maximum DSW size is set by the system designer, the current dynamic size of the DSW is computed by (7) where  $\beta$  and  $\gamma$  are computed using simultaneous equations (8). Fig. 2.(b) shows how the DSW works. When a node observes a bad behavior by one of its neighbors, the current Overall Trust for that neighbor gets lower. Thus, we want to consider more previous behaviors, so the DSW size is increased. In the figure we see on the top a window size of five with one bad behavior. Later, as represented in the window on the bottom of the figure, two more bad behaviors have occurred, so the DSW size has increased so that it can remember all three bad behaviors in the calculation of trust.

6) *Long-term Time-based Redemption*: Along with the SWs, we also employ a TBR that complements the disadvantages of BBR (Section II.A) that could occur in PT. Long-term Time-based Redemption (LTR) is used to eliminate behaviors that are too old to be considered relevant. While the SW uses the number of old behaviors to remove behaviors from consideration, LTR uses time. Every behavior that is accumulated for the computation of PT has a timestamp representing the observed time. A behavior will be ignored if

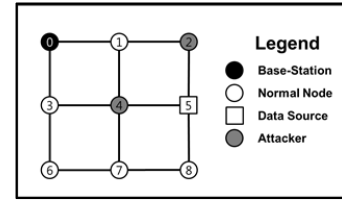


Figure 3. Topology

the timestamp is older than a specified time threshold. Thus, if the behavior information is too old to be considered, LTR ignores the behavior and calculates the PT without it. The LTR threshold is typically specified in orders of magnitude higher than the collection of behavior data. So if we collect hundreds of behaviors in an hour, the LTR threshold may be on the order of days.

#### IV. SIMULATION

In order to evaluate how well PT meets the criteria specified in Section I, we compared our trust redemption mechanism to several other redemption mechanisms. In this Section, first, we demonstrate how PT can prevent faulty detections, since the wireless network is not always stable, and thus may have some non-malicious transmission problems. Second, we show how PT can detect an On/off attack, and compare the performance to other redemption schemes. The network topology that we used in our simulations is based on a WSN with two Selective Forwarding nodes. We chose to test the On/off attack with a Selective Forwarding attack because it is relatively easy to observe how many packets are dropped during a transaction by various schemes [15][17]. In our simulations, we used the trust reporting mechanism of SARP.

All of our tests were run on TOSSIM [20], which was designed to ease the development and testing of WSN applications. The simulated nodes reported their current trust values and the routes they used to reach the BS. The DS reported the number of packets generated and the BS reported the overall throughput by comparing the number of delivered packets to the number of generated packets.

##### A. Topology

In our network topology, there are nine nodes, as shown in Fig. 3. Each node is identified by a number. The black solid lines show the communication status between nodes. If there is no black solid line between two nodes, they are not able to communicate with each other directly. When a node wants to deliver any information to another node that is not directly connected by a black solid line, it must communicate through another node by using send-receive-forward processes. The communication lines are created during the *Node Discovery* (see Section III.A.2.a). For our experiments, we set node 5, a three-hop node, as a DS, and it is marked with square. To test the efficiency of PT against Selective Forwarding (SF) and On/off Selective Forwarding (OSF), we located one SF attack node (node 4) and one OSF attack node (node 2) on the shortest paths from the DS to the BS.

1) *Attack nodes and possible paths*: The DS tries to deliver information to the BS through neighboring nodes. Node 4 is a

SF attack node that forwards only 70% packets in each transaction. Node 2 is an OSF attack node that sometimes forwards 10% packets and other times it forwards all packets. These nodes are marked with shaded circles in Fig. 3.

There are three possible three-hop paths. Since node 4 forwards only 70% packets, the BS always gets only 70% packets through the paths **5-4-1-0** and **5-4-3-0**. On the other hand, **5-2-1-0** delivers all packets most of the time, but sometimes delivers only 10% packet because node 2 is an OSF attack node.

When the malicious nodes are excluded from the routing path by a trust evaluation scheme, the DS delivers the information through one of the following detoured paths **5-8-7-4-1-0**, **5-8-7-4-3-0**, and **5-8-7-6-3-0**. These paths more likely consume more energy than the three-hop paths because they require five-hops. These paths may not be the best choice even if they deliver all packets in every transaction because they may require more energy.

During the simulations, the trust of nodes 2 and 4 became reduced because of the SF and OSF attacks. When the trust of a node was reduced low enough (below a specified threshold) the DS routed around the attack nodes and chose a longer path to the BS. If the chosen redemption scheme was able to recover the trust of nodes 2 and 4, they were again considered as routing options by the DS.

The topology that we used in our simulations is small, and does not represent the size of a real WSN. However, this small segment of a WSN is sufficient to demonstrate the efficacy of our PT redemption scheme. Since each node maintains trust evaluation and redemption information independently, the solution is scalable. The topology we chose for our simulations represents the small segment of a WSN surrounding attacking nodes.

### B. Redemption Schemes

As mentioned in Section II, trust redemption can be done in two ways; BBR and TBR. In order to recover trust, BBR needs to observe more good behaviors, while TBR periodically recovers the trust. In order to show how PT provides defense against On/off attack and how it is beneficial to the network, we compared PT to a BBR-based scheme and a TBR-based scheme, specifically TBDR from SARP (see Section II.A.1.b). Note that we did not compare our scheme directly to any of the related schemes mentioned in Section II because these redemption schemes were not designed specifically for WSN routing. Rather, we used the main concepts of BBR and TBR that have been proposed by those previous schemes.

1) *Behavior Based Redemption*: BBR recovers trust based on subsequent good behaviors. To implement a BBR-based scheme, we employed a trust calculation based on a Bayesian approach and it is calculated by (9) [18].

$$\begin{aligned} GFB_i &= GFB_i + NFP_i, & BFB_i &= BFB_i + NLP_i \\ FT_i &= \frac{GFB_i + 1}{GFB_i + BFB_i + 2}, & OT_i &= FT_i \end{aligned} \quad (9)$$

where the  $NFP_i$  represents the number of forwarded packets by node  $i$ , in a particular transaction, the  $NLP_i$  represents the number of lost packets by node  $i$ , in the transaction, and the  $FT_i$  represents the Forwarding Trust of node  $i$ . Recall that, for simplicity we are only concerning ourselves with a selective forwarding attack, so the Overall Trust is the same as Forwarding Trust. Also recall that we do not consider all behaviors, but use the sliding windows described in Section III to determine how many past behaviors to use in the computations.

On one hand, when the Overall Trust of a node is decreased to below the threshold of a network, the DS or FD will not choose the node anymore because a BBR scheme relies on subsequent behaviors to redeem trust. Therefore, the node will not be able to recover its trust because there is no further behavior to observe. On the other hand, if an OSF attack node attacks with appropriate good behavior ratio that is enough to keep the Overall Trust above the threshold, the OSF attack node would be able to attack the DS or FD continuously.

2) *Time Based Redemption*: TBR recovers trust periodically. In the TBR-based scheme that we implemented, we calculate the Overall Trust of a node by using (10).

$$\begin{aligned} GFB_i &= GFB_i + NFP_i, & BFB_i &= BFB_i + NLP_i \\ FT_i &= \frac{NFP_i + 1}{NFP_i + NLP_i + 2}, & OT_i &= FT_i \end{aligned} \quad (10)$$

and every six seconds, each node updates the trust of its neighboring nodes by using (11).

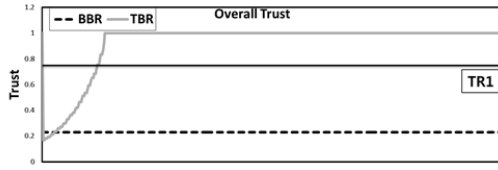
$$\begin{aligned} RF_i &= \left( \frac{GFB_i + 1}{GFB_i + BFB_i + 2} \right) + 1.0 \\ FT_i &= FT_i \times RF_i, & OT_i &= FT_i \end{aligned} \quad (11)$$

where the  $RF_i$  represents the Redemption Factor of the node  $i$  which controls the redemption speed depending on the current trust.

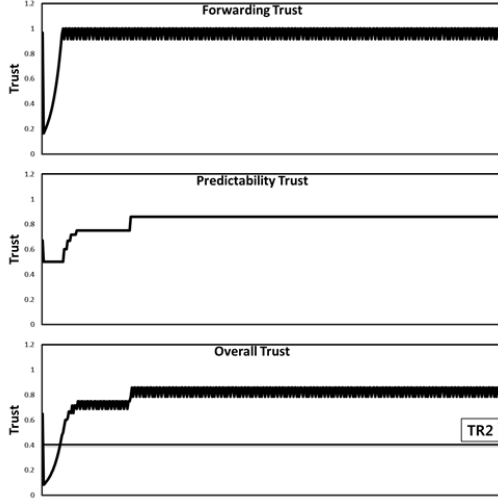
3) *Predictability Trust based Redemption (PTR)*: PT counts the number of the behaviors that did and did not satisfy the designer's expectation. In this paper, we only considered the 100% transaction as a good behavior. The Overall Trust of a node is calculated based on the forwarding behaviors and is shown in (12).

$$\begin{aligned} GFB_i &= GFB_i + NFP_i, & BFB_i &= BFB_i + NLP_i \\ FT_i &= \frac{NFP_i + 1}{NFP_i + NLP_i + 2} \\ GPB_i &= GPB_i + SB_i, & BPB_i &= BPB_i + DB_i \\ PT_i &= \frac{GPB_i + 1}{BPB_i + BPB_i + 2} \\ OT_i &= FT_i \end{aligned} \quad (12)$$

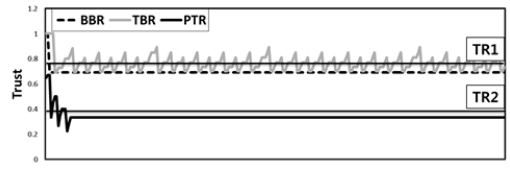
and every six seconds, each node updates the trust of its neighboring nodes by using (13).



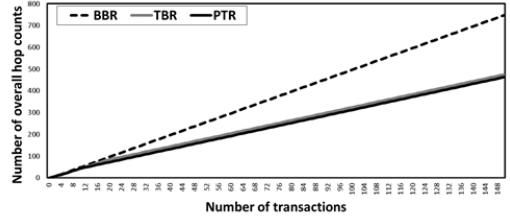
(a). The trust changes of node 2 in BBR and TBR.



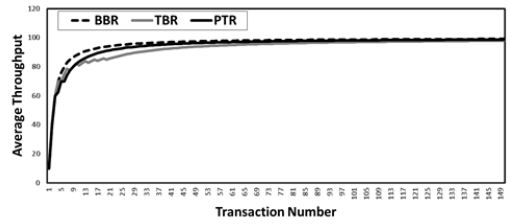
(b). The trust changes of node 2 in PTR.



(c). The trust changes of node 4 in BBR, TBR, and PTR.



(d). Overall hop counts for delivering the information.



(e). Average throughputs.

Figure 4. In case of a bad behavior was a temporary error.

$$RF_i = \left( \frac{GFB_i + 1}{GFB_i + BGB_i + 2} \right) \times PT_i + 1.0 \quad (13)$$

$$FT_i = FT_i \times RF_i, \quad OT_i = FT_i \times PT_i$$

where  $SB_i$  represents satisfied behavior of node  $i$  and  $DB_i$  represents disappointed behavior of node  $i$ . When the behavior satisfied the designer's expectation,  $SB_i$  is set to 1 and  $DB_i$  is set to 0. Otherwise,  $SB_i$  is set to 0 and  $DB_i$  is set to 1.

We have set the size of the FSW to be 5, the minimum size of the DSW to be 5, and the maximum size of the DSW to be 65. As a result of these settings, the PT will allow at least three bad behaviors and at best seven bad behaviors based on the results of tests we performed on various window sizes [18]. We also know that PT should be able to detect the six-good-one-bad On/off attack, and not detect an On/off attack with a higher ratio. During the simulation, the current size of the DSW was calculated using (14) which is deduced by using (8).

$$Size = \text{roundoff}((-66.66666667 \times PT) + 71.66666667) \quad (14)$$

The trust threshold for the routing scheme has to be set so that if a node's trust in its neighbor becomes too low, the node will not use that neighbor in its routing path. In our simulations we had to use two different trust thresholds. For BBR and TBR, we set the trust threshold to 0.75 (TR1), which is enough to detect the malicious behaviors of the node 4. For PTR, we used a threshold of 0.4 (TR2) because if we used 0.75 as we did with the other redemption schemes, PTR isolates node 4 too soon. PTR is more sensitive and would not be able to recov-

er trust to an acceptable level above the 0.75 threshold before the PT value brings it down again. Note that using a lower threshold for PT in our simulations means that it is enough to detect the SF attack node while the other redemption schemes require a higher threshold. This makes it possible to design a more strict system than other redemption schemes by making the threshold higher.

### C. Simulation Results

In our simulations, the DS initiates 150 transactions that each deliver 10 packets to the BS. For each test we present the trust changes, the overall hop counts for delivering the packets, and the average throughput for each redemption scheme.

1) *Temporary Error*: In this first test, we demonstrate why trust redemption is important in WSNs. We assumed that node 4 was a SF attack node that forwards only 70% packets. Node 2 was not an OSF attack node but dropped 90% packets only once. This test was meant to simulate a situation in which a node had a temporary error, and was not malicious.

a) *Trust changes*: As we discussed in Section II, BBR is not able to recover trust if there is no further good behavior to observe. In this test, the BBR isolated node 2 from the network after one bad behavior, and the trust was not recovered as shown in Fig. 4.(a) because the DS did not use the node again even if the bad behavior was the only error made by that node. Moreover, the BBR isolated node 4, which is a SF attack node, because its forwarding rate was below the trust threshold (TR1 in the Fig. 4.(a)). Thus, the DS was not able to use the three-hop paths, and instead was forced to use the longer six-hop detoured path. On the other hand, TBR and PTR, which are able to recover the Forwarding Trust based on



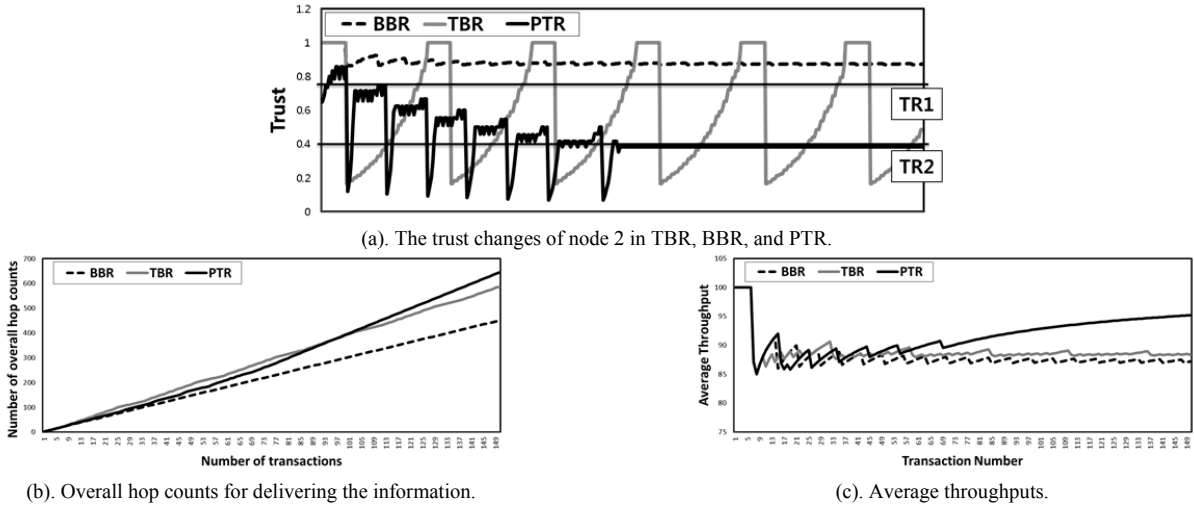


Figure 5. In case of six-good-one-bad On/off Selective Forwarding attack

time, show different results from the BBR. Fig. 4.(a) also shows the trust changes of the Overall Trust of the node 2 for TBR. After the temporary error, TBR provided a second opportunity to node 2, and it made the three-hop path available again. Fig. 4.(b) shows the trust changes for PTR. Note that the Forwarding Trust dipped below the threshold when the bad behavior was observed. At that same time, the PT went down because the bad behavior was unexpected. Consequently the Overall Trust, which is computed using Forwarding Trust and PT also had the initial dip in its value. Because this was a one-time error, the Forwarding Trust was able to recover, and the PT also recovered because no other unpredictable bad behaviors occurred. We can also see that because Overall Trust is computed using PT that the maximum value of Overall Trust was limited by the value of PT. Since the Overall Trust recovered above the TR2, the three-hop path, which uses node 2, became available again. Further, Fig 4.(c) shows that PTR isolated node 4, which is a SF attack node, from the network after three transactions while TBR was attacked by node 4 every time the trust recovered. Note that the Overall Trust for PTR in Fig 4.(c) was calculated like the Overall Trust for PTR in Fig 4.(b).

*b) Hop counts:* When the BBR isolated nodes 2 and 4 from the network because of the one-time bad behavior, the hop count for delivering the packets was increased more than the hop count for TBR and for PTR which both gave another opportunity to node 2. Fig. 4.(d) illustrates this difference. As we discussed in Section I, the hop count can affect the network lifetime and transaction time significantly if encryption is employed to defend against external attacks. Our simulation was based on nine nodes, but in a more realistic larger WSN, the difference between the hop counts would be much larger. This means the network consumes much more energy and the transaction time would be longer due to this one temporary error.

*c) Average throughput:* Although BBR consumed more energy and spent more transaction time, the average throughput was not much different than for TBR and PTR, as

shown in Fig. 4.(e). This is because BBR finds a reliable path to the BS, and thus the throughput approaches 100% after the one bad behavior. Although the TBR and PTR were attacked by node 4 a few more times as it waited for node 2's trust to recover, the average throughput quickly approached the throughput of BBR.

The results of this set of tests demonstrate that with just one anomalous bad behavior, PTR is able to provide almost optimal throughput (as does BBR) and use minimal number of hops (as does TBR).

*2) On/off attack:* In this Section, we discuss the benefit of isolating an OSF attack node from the network. In this test, node 2 attacked the DS with six-good-one-bad OSF attack, which is enough to disturb the other redemption schemes.

*a) Trust changes:* As we discussed in Section I, an On/off attack is designed to disturb a redemption scheme by making the scheme think that one bad behavior is anomalous, and thus recovering a node's trust value. When a redemption scheme is disturbed, the malicious node is able to attack the target node continuously. As shown in Fig. 5.(a), the trust of node 2 in BBR was never decreased to the below the TR1. Since the sliding windows always had five more good behaviors than bad behaviors, (9) converged around 0.86. Thus, the DS could not detect the OSF attack node because this value was always greater than the threshold. On the other hand, TBR recovered the trust of node 2 above the TB1 continuously. This is because whenever there was a bad behavior that would reduce the trust, there was enough time to recover the trust above the threshold. Neither BBR nor TBR was able to take into account the accumulation of damage caused by previous bad behaviors. The trust in PTR was decreased below the TB2 threshold after seven bad behaviors. The use of PT allowed the DS to not only recover more slowly as the bad behaviors accumulated, but also recover to a lower maximum Overall Trust value, which was eventually below the TB2 threshold. Once the maximum Overall Trust value was below TB2, it was not able to recover further.

*b) Hop counts:* In Fig. 5.(b), the BBR showed the lowest

number of overall hop counts, and the PTR showed the most number of overall hop counts. This is because BBR was not able to keep the trust of the OSF attack node below the TR1 threshold, and therefore it was considered a viable routing choice, allowing the DS to use the shortest path. On the other hand, the PTR isolated the OSF node after seven bad behaviors, so it started using one of the detoured paths. These results require a designer to decide a trade-off between the overall hop count and the average throughput. This trade-off is discussed in [18]. Because TBR produced a repeating pattern of trust, it alternated between choosing a three-hop path and a six-hop path. This is why the TBR hop count line in Fig. 5.(b) is between the BBR line and the PT line.

c) *Average throughput*: Since the BBR and TBR were attacked by the OSF attack node continuously, the damages were accumulated. Fig. 5.(c) shows that the accumulated damages affect the average throughputs. The average throughput of the BBR converged around 87.2% (damage rate: 12.8%), and the TBR was converged around 88.6% (damage rate: 11.4%) while the PTR average throughput approached 100% because it is the only scheme that was able to eliminate the malicious node from the routing path. At this point, the designer needs to decide one of the advantages. First, the BBR is able to maintain the shortest path, but it would be able to send only 87% packets. Second, the PTR ultimately is able to send 100% packets but it needs to use the detoured path.

## V. CONCLUSION AND FUTURE WORKS

Predictability Trust is a concept that allows for considering accumulated previous behaviors to compute trust of a node in a network. While we have demonstrated PT in a secure routing scheme for WSNs, it is a notion that can be applied to any collaborative network in which entities need to trust each other in order to work together. When a smart attacker knows that the collaborating entities allow for trust redemption, the attacker can try to plan attacks in such a way to fool the system into regaining trust after time or after subsequent good behaviors. With the use of Predictability Trust, the previous behaviors of this attacker will make a collaborator more wary of working with it again, and will redeem trust more slowly. If the accumulated behaviors prove to be bad enough, the collaborator will choose not to work with the attacker again.

As we mentioned previously, PT can allow for a flexible design in which a system designer is able to decide the number of minimum and maximum opportunities for a node, and to determine an acceptable level of On/off ratio to trade-off security for performance. In future work we plan to implement PT in other types of collaborative networks such as social networks and cloud computing environments.

## REFERENCES

- [1] F. Hu and N. Sharma, "Security considerations in ad hoc sensor networks," *Ad Hoc Networks*, vol. 3, pp. 69-89, 2005.
- [2] Y. Hu, A. Perrig, and D. Johnson, "Ariadne: A secure on-demand routing protocol for ad hoc networks," *Wireless networks*, vol. 11, pp. 21-38, 2005.
- [3] Y. Hu, D. Johnson, and A. Perrig, "SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks," *Ad Hoc Networks*, vol. 1, pp. 175-192, 2003.
- [4] P. Papadimitratos and Z. Haas, "Secure routing for mobile ad hoc networks," *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference*, vol. 31, pp. 193-204, 2002.
- [5] S. Yi, P. Naldurg, and R. Kravets, "Security-aware ad hoc routing for wireless networks," *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pp. 299-302, 2001.
- [6] D. Carman, P. Kruus, and B. Matt, "Constraints and approaches for distributed sensor network security (final)," *DARPA Project report, (Cryptographic Technologies Group, Trusted Information System, NAI Labs)*, vol. 1, pp. 1, 2000.
- [7] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," *Ad Hoc Networks*, vol. 1, pp. 293-315, 2003.
- [8] A. Pathan, H. Lee, and C. Hong, "Security in wireless sensor networks: issues and challenges," *Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference*, vol. 2, pp. 6-1048, 2006.
- [9] J. Lopez, R. Roman, I. Agudo, and C. Fernandez-Gago, "Trust management systems for wireless sensor networks: Best practices," *Computer Communications*, vol. 33, pp. 1086-1093, 2010.
- [10] S. Buchegger and J. Y. Le Boudec, "Self-policing mobile ad hoc networks by reputation systems," *Communications Magazine, IEEE*, vol. 43, pp. 101-107, 2005.
- [11] S. Buchegger and J. Le Boudec, "Performance analysis of the CONFIDANT protocol," *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pp. 226-236, 2002.
- [12] P. Michiardi and R. Molva, "Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks," *Proceedings of the IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security: Advanced Communications and Multimedia Security*, pp. 107-121, 2002.
- [13] S. Bansal and M. Baker, "Observation-based cooperation enforcement in ad hoc networks," *Arxiv preprint cs/0307012*, 2003.
- [14] Q. He, D. Wu, and P. Khosla, "SORI: A secure and objective reputation-based incentive scheme for ad-hoc networks," *Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE*, vol. 2, pp. 825-830, 2004.
- [15] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," *Proceedings of the 6th annual international conference on Mobile computing and networking*, pp. 255-265, 2000.
- [16] J. Newsome, E. Shi, D. Song, and A. Perrig, "The sybil attack in sensor networks: analysis & defenses," *Proceedings of the 3rd international symposium on Information processing in sensor networks*, pp. 259-268, 2004.
- [17] L. C. DiPippo, Y. L. Sun, and K. Rhan, Jr., "Secure Adaptive Routing Protocol for Wireless Sensor Networks," 2010, Technical Report – TR10-329.
- [18] A. Jøsang and R. Ismail, "The beta reputation system," *Proceedings of the 15th Bled Electronic Commerce Conference*, pp. 41-55, 2002.
- [19] Y. Chae, "Redeemable Reputation based Secure Routing Protocol for Wireless Sensor Networks," Master of Science, Department of Computer Science and Statistics, University of Rhode Island, 2012. Tech Report: 12-331
- [20] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire TinyOS applications," *Proceedings of the 1st international conference on Embedded networked sensor systems*, pp. 126-137, 2003.