$PLUS$: Parameterized and Localized trUst management Scheme for sensor networks security

Zhiying Yao
Electronics Telecommunications Research Institute
Korea
Email: yaozy@etri.re.kr

Daeyoung Kim
Information and Communication University
Korea
Email: kimd@icu.ac.kr

Yoonmee Doh
Electronics Telecommunications Research Institute
Korea
Email: ydoh@etri.ac.kr

***Abstract*— The wireless and resource-constraint nature of a sensor network makes it an ideal medium for attackers to do any kinds of vicious things. In this paper, we describe $PLUS$, a parameterized and localized trust management scheme for sensor networks security, where each sensor node maintains highly abstracted parameters, rates the trustworthiness of its interested neighbors to adopt appropriate cryptographic methods, identify the malicious nodes, and share the opinion locally.**

Results of a serious of simulation experiments show that the proposed scheme can maximize security as well as minimize energy consumption for sensor networks. And also, the secure routing proposed based on $PLUS$ shows its benefit and feasibility.

I. Introduction

Ubiquitous computing, regarded as "the calm technology that recedes into the background of our lives", opens up various novel applications. It relies on tiny devices embedded in everyday objects and environments, intelligently collecting and delivering information and communicating without any fixed infrastructure. Sensor networks as a sufficient enabling technology can be used as room temperatures controller, as real-time traffic monitor; and can provide security in office buildings, military surveillance, etc.

Security is extremely important when a sensor network deployed in a hostile environment. The sensitive data must be well-protected to ensure information authenticity, confidentiality and integrity. In some military applications, the highest level security must be provided, otherwise, the result could be extremely dangerous.

Existing works done here mainly rely on cryptographical schemes (e.g. INSENS [1] and SPINS [5]), however, they still suffer from many security vulnerabilities, such as node-capture attacks and denial-of-service attacks. New mechanisms are needed to address this concern.

In this paper, we focus on solving sensor network security with the help of efficient trust management mechanism ([3]). Deriving trustworthiness statistically can make security stronger, simpler, and more efficient. The constructive features of $PLUS$ – a Parameterized and Localized trUst management Scheme are as follows:

- Maintain a parameter database to depict the operational environment, application types, network status, and node local information.
- Construct a shared library to provide common services used by other components.
- Design four logical components: network I/O, which deals with the network traffic; routing operator, which provides corresponding packet handles; trust estimator, which rates nodes' trustworthiness; and security responser, which manages inside of $PLUS$ from security perspective, and takes local actions.

We then design a secure routing protocol as a $PLUS$ application to provide secure communication in the established trustworthy environment.

The remainder of the paper is organized as follows: Section II reviews the related research efforts. Section III describes the proposed $PLUS$ with a detail description about each component. The performance study through simulation is conducted in Section IV. Section V gives an application based on $PLUS$, a secure routing protocol. The paper concludes in Section VI.

II. Related Work

We investigate two classes of work closely related to the paper: previous security solutions for sensor networks, and existing works in building trust model.

A. State of The Art in Security for Sensor Networks

Due to the unique characteristics of sensor nodes, the wired network security solutions may not be ap-

1-4244-0507-6/06/$20.00 ©2006 IEEE

plicable for wireless sensor networks. Cryptographical, hashing mechanisms have to be utilized with certain modifications. We will review the current issues from four aspects: intrusion prevention, intrusion tolerance, intrusion detection, and key management.

From an intrusion prevention perspective, Perrig et al design security protocols- SPINS- for sensor networks [5], which consists of Sensor Network Encryption protocol (SNEP) and μTESLA (the "micro" version of TESLA). As an early and important result in sensor network security field, the main aim, the authors want to deliver, is the feasibility to provide security to resource-constraint tiny senor nodes with symmetric cryptography mechanisms. However there are many drawback of this strategy, such as the lack of schemes to prevent against the node-capture attacks and denial-of-service attacks. And also, source routing, which is too expensive in terms of node state and packet overhead, may not be appropriate to sensor networks.

For intrusion tolerance, [1] and [2] propose multiple paths routing approach to provide fault tolerance. One important property of this approach is that even though a malicious node may be able to compromise a small number of nodes in its vicinity, it can not cause widespread damage in the network. However, one could argue that if the region is secure enough, the approach will waste too much overhead to redundantly build disjoint paths and blindly transmit packets with multiple paths.

The fatal denial-of-service attacks are particularly vulnerable to a sensor network due to its wireless medium, changeable topology, cooperative working pattern, and limited resource. Many of the intrusion detection techniques developed on a fixed wired network are not applicable in this new paradigm. Some works are done for ad hoc networks. For instance, [9] propose a solution to install intrusion detection agents on every node to cooperate together towards the detection. However, the approach is not well suited to the energy efficiency requirement of wireless sensor networks; does not specify how the IDS architecture interact with the underlying key management and communications protocol. Most importantly, how much resources (in terms of computing power, memory and energy) required for sensors to support intrusion detection remains unanswered.

Obviously, the existing security solutions are not working very well. What is missing is a trustworthiness evaluation scheme that can give individual nodes ability to efficiently estimate their local environment and then to make appropriate decision locally.

B. State of The Art in Trust Management

All kinds of transactions, interactions, and communications in human life are based on trust. People always think about trust when they handle affairs, sometimes, unconsciously. So do the sensor networks. In sensor networks, one single node can not do anything. Instead, they must co-work to accomplish higher level tasks. Therefore, they also need trust.

Generally speaking, there are two main trust models in the literature, centralized trust model and distributed trust model. In the formal, a particular trusted intermediary, called 'Trust Authority' (TA), is used to form trust relationships [4]. Although the implementation is efficient and manageable, it suffers from robustness and scalability problems. Yahalom et al [8] discussed in detail the concept of trust in distributed systems. They highlighted the fact that trust requirements in security protocols are necessary. Consequently, they defined trust classes, made the distinction between direct and recommendation trust and proposed a formalism for analyzing trust in authentication protocols. However, their work falls short of defining the acquisition of extended trust information.

[5] designates base station as the central trusted authority in protocol design. Since sensor networks are a type of dynamic ad hoc network with large-scale sensor nodes, this kind of design suffers from the problematic central failure and scalability limitation. Besides, a sensor node mainly cares about the trustworthy of its neighboring nodes due to the multi-hop transmission nature. Therefore, the distributed trust model is more suitable for sensor network security design. With the point in mind, we intend to define a more sensor network oriented trust model as the basic for our scheme.

III. Parameterized and Localized trUst management Scheme

In this section, we introduce our proposed Parameterized and Localized trUst management Scheme ($PLUS$) in detail. The node architecture used to implement the scheme is shown in Figure 1. It is designed with the consideration of the low cost sensor nodes. The key design point is to establish secure sensor networks with the highly abstracted parameters and localized trust management mechanism.

A. Parameter Database

The reason of using parametrization is to quantify, mange, visualize and exploit the abundant available information simply. Besides, storing decision-making parameters is much more memory-saving than using

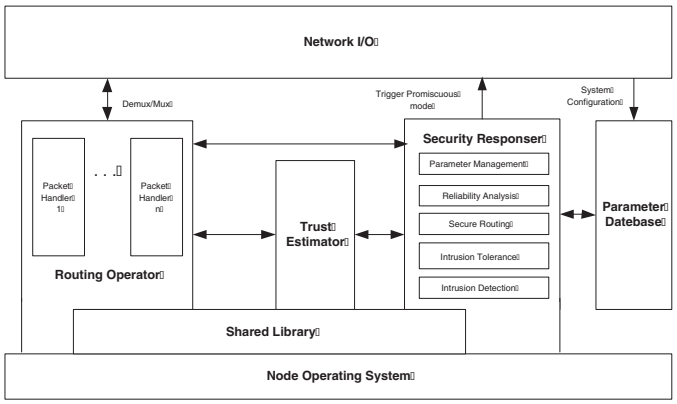

Fig. 1: $PLUS$ Architecture

lengthy describing code, and can make the scheme more adaptive and extensible.

Broadly speaking, it includes two categories: infrequent-changed parameters, which are defined for operational environment (benign or hostile), application types and the ones having closed relation with the above two, such as some trust related attributes (e.g. regulated trust level and thresholds); and frequent-changed parameters, which are defined for network status (e.g. network connectivity), and node local information (e.g. resource availability and neighbor trustworthiness). Herein, we pick up part of second class parameters which describing the information about one of the local node's neighbors to show the possible structural list. We let the unit parameters are represented by single values, and the union parameters, which consists of interrelated unit parameters, are marked starting with character "&". Note that, the union parameters can be also constructed recursively. And with the follow-up explanation, the parameters here will be more clear.

```
&neighborInfor = (
    &neighborDescriptor = (
        neiID
        neiLocation
    )
    &neighborRelation = (
        //parent towards base station
        //parentAlternative used once parent fails
        //child deviates from base station

    neiDuty =< parent, child, parentAlternative >
        //dedicated key with the local node neiKey
        //Defined by adopted routing protocol
        routingMetric
    )
    &neighborTrust = (
        //for computing T_po
        latestCommTimestamp
        //for computing T_re
        noOfReplyFromBS
        //for computing T_coo
        noOfCorrectForward
        neiTrustValue
    )
)
```

B. Trust Estimator

As explained, an individual senor node should have the reasonable security-sensitive ability to evaluate its local site, identify the failed neighbors timely, and take intelligent actions to carry out its network duty. We construct the trust estimator to achieve the goal, which is based on a localized trust model, shown in Figure 2.

For one thing, we give some definitions: the node, which performs evaluation, as judge; the node, which is in the radio range of the judge and will be evaluated, as suspect; and the node, which maintains the trust value of the same suspect with the judge and sends out the corresponding opinion periodically or intentionally as jury. Besides, the trust relationship should have two characteristics: it is not symmetric, that is, if A trust B, B maybe not trust A, where A and B are mutually neighboring nodes; and, it is time-evolving, which needs to be updated upon receipt of new interactive communication or new recommendations.

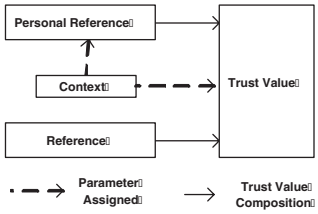

Fig. 2: Conceptual Distributed Trust Model

Next, let us go into depth of the distributed trust model, which bears two salient features, recommendation-based trust and trust-based recommendation. The judge, who wants to achieve a comprehensive trustworthy of the suspect, requires not only personal reference, but reference, that is, recommendation-based trust. Personal reference is derived from the direct interaction with the suspect, which can be the packet, originated or forwarded from the suspect; or can be the observed behavior of suspect, such as forwarding correctness. Reference is obtained via combining the recommendation provided by the juries. Actually, the recommendation is the personal reference of certain jury towards the suspect. Note that, the trustworthiness of juries has to be taken into account to against malicious use, so called trust-based recommendation. Then, we will explain the logical process to compute the corresponding values in detail.

1) Personal Reference: We broadly classify the parameters used for obtaining personal reference to two types as shown in Figure 3: parameters about cryptographic operations, which represent the security mechanisms used in the communication, and can disclose attacks (e.g. message forgery and modification); and parameters about nodes' interactive behavior, which can

reflect nodes availability, and reveal attacks (e.g. dropping and denial-of-service). Then we can quantify the personal reference of judge j about suspect i .

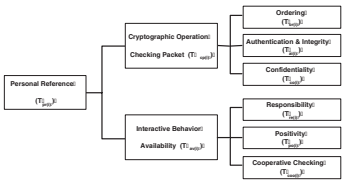

Fig. 3: Personal Reference Parameter Tree

For cryptographic mechanisms, herein, we consider a common case, where encryption/ decryption and integrity checking are used. And besides, identity of base station is authorized.

a) $T_{or(i)}$*:* indicates whether the packet, forwarded by suspect i, is from base station and whether it is a fresh information. Since important control packets are always sent by the base station, the identity of base station must be verified. We use hash sequence number (HSN) with the characteristic of hash function . Whenever the base station originates packet (e.g. topology discovery command), it will append a fresh HSN in the outgoing packet. When a node receives this kind of message, it will check the new HSN to evaluate the suspect's behavior (usually, it will be the upstream node). The detailed evaluation procedure is shown in Algorithm 1. Note that, in some case the packet originated by the base station will traverse only a certain subset of the network (e.g. selective forwarding of packets), and packet loss will be caused by other physical problem, application designers have to estimate a practical Ω, and design an appropriate decay function.

b) $T_{ai(i)}$*:* indicates whether the received packet is modified on the way by checking the message authentication code (MAC) with the corresponding key. The result can be expressed as:

$$T_{ai(i)} = \begin{cases} 1 & \text{MAC match} \\ 0 & \text{else} \end{cases} \quad (1)$$

c) $T_{co(i)}$*:* indicates whether the ciphertext received can be decrypted to meaningful plaintext with the corresponding key. The result can be obtained as:

$$T_{co(i)} = \begin{cases} 1 & \text{meaningful plaintext} \\ 0 & \text{else} \end{cases} \quad (2)$$

d) $T_{cp(i)}$*:* indicates the correctness of the incoming packet from suspect i. Note that a node can change its behavior over time, we concern only its current status.

$$T_{cp\,(i)} = T_{or\,(i)} \times (T_{ai\,(i)} \quad T_{co\,(i)}) \quad (3)$$

In sensor networks, nodes have to cooperate together to perform network functions. Nodes, which are selfish to drop the forwarding packets or mount denial-of-service attack to send out duplicated packets, have to be disclosed and isolated. Hence, we define following parameters to abstract nodes' interactive behavior.

Algorithm 1 Algorithm for computing $Tor(i)$

when judge j receives a packet from suspect i
if $hsn_{t_1} = H^1(hsn_{t_2})$ **then**
  $T_{or}(i) = 1$ *// from BS, and with correct order*
  exit
else
  $\Omega = \delta(t_2 - t_1)$
  $k = 2$
  while $k < \Omega$ **do**
    compute $H^k(hsn_{t_2})$
    if $hsn_{t_1} == H^k(hsn_{t_2})$ **then**
      compute $F_{(k)}$
      $T_{or}(i) = F_{(k)}$ *// from BS, but $(k-1)$ packets lost*
      exit
    else
      $k++$
    end if
  end while
  $T_{or}(i) = 0$ *// a spurious packet*
end if
END
Notations
F : a decay function with field $(0, 1)$, which is a punitive measures towards i due to packet loss. For Example, we can use $e^{-\lambda k}$
H : a hash function
H^k : a times operation with H
δ : the times that H can be executed in one unit time interval.
t_1 : the time previous packet received from i
t_2 : the time current packet received i
Ω : the maximum operation times of H between t_1 and t_2
hsn_{t_1}: the corresponding one stored in judge j
hsn_{t_2}: the HSN appended in the current packet

e) $T_{re(i)}$*:* indicates whether judge j can receive reply from the base station through node i, once the judge alarms or asks for some information to base station, such as shared key update. We concern a statistic value, between 0 and 1, expressed as:

$$T_{re\,(i)} = \frac{\text{number of replies}}{\text{number of requests}} \quad (4)$$

Note that, we also consider the correctness of the reply. Either nodes' malicious behavior or packet loss (i.e. caused by network congestion or wireless interference) represents a distrust situation and will decrease the value.

f) $T_{po(i)}$*:* is used for checking nodes' positivity by base station, especially in continuous [7] sensor network. It also can be used by an individual node to check positivity of suspect i, that is, whether the node can participate in the exchange of opinions and whether the node can report measurement to base station with appropriate frequency. The parameter can help judge to disclose a selfish suspect and a denial of service attacker. To obtain the value, the application designer designates the interval [min, max] to represent the supposed intercommunication times between network elements in a unit time interval. And judge can compute the actual times of intercommunication between itself and the suspect,

notated as $times_{(i)}$. Then we can compute $T_{po(i)}$ as:

$$T_{po(i)} = \begin{cases} 1 & times_{(i)} \quad [\min,\ \max] \\ \frac{times_{(i)}}{\min} & times_{(i)} < \min \\ \frac{times_{(i)}}{\max} & times_{(i)} > \max \end{cases} \tag{5}$$

Note that, the above two parameters are defined considering different types of applications.

g) $T_{coo(i)}$*:* is a result obtained by sniffing in promiscuous mode. Considering the excess energy consumption and low-cost nature of sensor nodes, it is only used when the above two parameter values are abnormal by comparing with the thresholds provided. The judge routes some packets via suspect and monitors whether the suspect can forward them correctly in certain time period. It can be obtained by a forwarding ratio fr and a decay function $D_{(x)}$, where $x, D_{(x)} \in [0, 1]$.

$$fr = \frac{\text{number of packets actually forwarded}}{\text{number of packets supposed to be forwarded}} \tag{6}$$

$$Tcoo(i) = \begin{cases} D(fr) & fr < 1 \\ D(1/fr) & fr \quad 1 \end{cases} \tag{7}$$

h) $T_{av(i)}$*:* is a value that indicates the availability of suspect i, and is integrated from the above three values. There are two points has to be considered to obtain the value:

- $T_{coo(i)}$ should be a more accurate value, but as we regulated, the promiscuous mode is so expensive that is triggered only if the other two values becoming abnormal.
- Since the characteristic of nodes' behavior depends upon the different application types, herein, we recur to a set of weighted values to adjust the appropriate proportion between $T_{re(i)}$ and $T_{po(i)}$.

Then, it is expressed as:

$$\begin{gathered} T_{av\,(i)} = T_{po\,(i)} \times W_{po} + T_{re\,(i)} \times W_{re} + T_{coo\,(i)} \times W_{coo} \\ where \quad W_{po} + W_{re} = 0.5 \quad and \quad W_{coo} = 0.5 \end{gathered} \tag{8}$$

At last, we can compute the personal reference value through a weighted summation of the two types of parameters:

$$\begin{gathered} T_{pr\,(i)} = T_{cp\,(i)} \times W_{cp} + T_{av\,(i)} \times W_{av} \\ where \quad W_{cp} + W_{av} = 1 \end{gathered} \tag{9}$$

2) Reference: As a kind of recommendation provided by the juries, it can help the judge obtain more comprehensive opinion on certain suspect.

To obtain reference, we propose a set of recommendation protocols to specify how the exchange of trustworthy information between the judge and jury happens. The basic performance requirement is energy efficiency. Herein, we list two assumptions: the network is density enough that each node has several juries surrounded; and the reference counted in must be sent from the jury whose trustworthiness is in the record.

We will not let individual sensor node periodically broadcast exchangeable trust information, which seems to be a bit "blindness-centric" and has the difficulty in finding the appropriate periodical time interval. Instead, our recommendation protocol is only triggered when the trust level of suspects changes (say, from level 4 decrease to level 3). The protocol includes two components: an active protocol, which is initialized by judge (say, A) for requesting trustworthiness of an interested suspect (say, B); and an anti-active protocol, which is also initialized by judge, but for informing problematic suspect (say, $B*$) with observed misbehavior specified. Both of them are controlled by the security responser component of the node A according to the trust policy. Actually, each judge also is another judge's jury. It is a relative identity.

To start the active protocol, node A first broadcasts an *exchangeRequest* packet to its adjacent neighbors. The packet includes the identifier of node B, it wants to evaluated, and its own identifier. Upon receiving the request, except node B, other neighbors of node A, which have trust values of node B, send an *exchangeReply* packet to node A, including the identifier of node B, the identifier of its own, and the provided trust value of node B. After collecting all the replies from neighbors, controlled by a timer started immediately after sending *exchangeRequest*, the node A discards the packets from not-known neighbors (i.e. the ones which are not in the parameter database of node A), and regards remain nodes as valid juries. Then, node A will calculate the reference as equation 10 shown. Note that, sequence number field can be used here for avoiding replay attack. We will not detail this point here.

To start the anti-active protocol, node A first broadcasts the *exchangeInform* packet towards its neighbors. The packet includes the identifier of node $B*$, the error code of misbehavior (e.g. drooping, silence), and its own identifier. Upon receiving the announcement, except node $B*$, other neighbors, which have trust values of node $B*$ check their own estimation value. Based on the trust policy, the neighboring node sends out its opinion: *exchangeAck* in case of agree with the judge, *exchangeArgue* in case of disagree with the judge. The former packet can simply include the ID of node $B*$ and ID of its own; and the latter packet can be same with the *exchangeReply* packet except the packet type. After collecting all the replies from neighbors, the node A discards the packets from not-known neighbors (i.e. the ones which are not in the parameter database of node A), and regards remain nodes as valid juries. Then, node A will reevaluate the reference of node $B*$ as

equation 10 shown only if most neighbors are against the announcement. Note that, sequence number field can be used here for avoiding replay attack. We will not detail this point here.

Because a judge can receive recommendation from not only one jury, we have to combine all the opinions together to get the final reference. Therefore, we propose a simple combination strategy, which is a kind of trust-based recommendation with consideration of jury's trustworthiness in order to against malicious use. Four adjustment factors corresponding to four trust level are assigned to give a proportion to the final result. They are represented as AF_l where $l = 1, 2, 3, 4$ and $AF_l \in (0, 1)$. So, for m juries, who have opinions on suspect i, the judge can compute the reference as:

$$T_{r(i)} = \frac{\sum_{k=1}^{m} AF_{lk} \times T_{(i)k}}{m} \tag{10}$$

3) Context: The module is responsible for maintaining the adjustable weighted values, which are passed from parameter database and will be used in personal reference and trust value computation process.

4) Trust Value: Now, the judge can obtain the trustworthiness of suspect i by weighted summation between the personal reference and reference. The result can be expressed as:

$$\begin{array}{c} T_{(i)} = T_{pr\,(i)} \times W_{pr} + T_{r\,(i)} \times W_r \\ where \quad W_{pr} + W_r = 1 \end{array} \tag{11}$$

C. Network I/O

The component is responsible for receiving packets, and directing them to routing operator based on the mapping between packet type and packet handle; for sending packets to next hop by checking *neiNetworkDuty*; for controlling the rate of incoming and outgoing packets specified by *incomingRate* and *outgoingRate* parameters; and for monitoring certain suspect's behavior requested by security responser. Additionally, network I/O has the ability to discard packets directly by checking the *blackNodeList* parameter.

D. Routing Operator

The component provides several packet handles corresponding to different packet types. Each packet handler has the ability to either decompose incoming packets or format outgoing packets, or both depending on the different packet type. Herein, we only focus on the security operations processed on the packets. With the consideration of minimizing the extra communications and computation overhead brought by security processes, we deliver different security levels to different packet types, modified from [6], which defines a corresponding security mechanism to each type of data (i.e. mobile code, location information, and application specific information). We define three necessary security levels for our packet types, shown as Table II.

TABLE I: Security Level Regulation Table

Security Level	Security Operations
I	OWS & ENC & MAC
II	ENC & MAC
III	ENC

And we assign the three different security levels to different packet types:

1) Security level I: reserved for *protocolPacket*, which is about topology construction and only initialized by base station, but has little payload.

2) Security level II: dedicated for *sensingPacket*, which carries sensed data and is transmitted from individual node to base station.

3) Security level III: used by *trustRelationExchangePacket*, which is mostly communicated between adjacent nodes. Herein, we omit MAC operation because it is only one-hop communication and MAC field increases the packet length.

E. Security Responser

The component integrates the mostly existing security research directions in sensor networks (e.g. reliability analysis, secure routing, intrusion detection, and intrusion tolerance), and can be performed more efficiently.

1) Parameter management policies: $PLUS$ has amount of parameters, which come either from the updatable theoretical calculation or human settings. Among them, some are prone to change. Therefore, a set of policies are required to regulate the alteration, and trigger corresponding actions once needed.

Herein, we mainly focus on the management of trust values. Firstly, we define a manner to regulate the discrete trust level for achieving a clear criterion to reflect the trust relationship. There is no 100% absolute trustworthy or untrustworthy exists, because of the dynamic nature of sensor networks and the unreliable wireless communication link. And considering application-dependent, there is no one universal value system. Therefore, we give the generalized trust categories as Table II, which can be configured upon the different targeted application characteristics. Take a military application for example, where higher security requirements should be met, we should expand the range of very low and low trust level properly with an aim at increasing vigilance.

TABLE II: Trust Level Regulation Table

Trust Level	Name	Description	Trust Value
1	Distrust	Untrustworthy	$(0, r_1]$
2	Minimal	Low Trust	$(r_1 r_2]$
3	Average	Common Trustworthy	$(r_2, r_3]$
4	Good	Trustworthy	$(r_4, 1]$

Then, we establish a lightweight trust evolution algorithm with the assumption that every node trusts its neighbors initially:

- If $T_{(i)}$ is within trust level 4, the local node can keep using the node.
- If $T_{(i)}$ is changed from trust level 4 to trust level 3, the local node has to trigger the promiscuous mode if the reduction is caused by the problematic interactive behavior; or trigger the anti-active protocol.
- If $T_{(i)}$ is decreased to trust level 2, the local node has to trigger the anti-active protocol. If the final trust value calculated is still staying in level 2, the node will be marked as black node, and there is no more punitive calculation processed for it unless encouraging calculation.
- If trust level is directly jumping to level 1, the local node has to trigger the active protocol. If the final trust value calculated is still staying in level 1, the node will be marked as black node, and there is no more punitive calculation processed for it unless encouraging calculation.

And security responser also compares certain parameters with their corresponding thresholds, such as the one used to compute suspect's positivity, to select appropriate operation. Besides, considering highly density network, there maybe large numbers of neighbors around a node. Hence, a likely upper bound of amount of neighbor entry has to be assigned based on memory size. Consequently, certain algorithms are needed to insert new neighbors and delete old or bad neighbors. Herein, we will not go in depth.

2) Usage of Trust Value: Through the result of managing parameter database, security responser component has ability to analyze neighboring nodes' behavior, adopt appropriate cryptographical mechanisms, select appropriate next hop based on trustworthiness, trigger monitoring mechanisms to detect malicious behavior, and properly adopt redundancy to achieve intrusion tolerance. We illustrated the benefit of our scheme by applying it to a typical example sensor network described in [10]. Additionally, we will give a more detail application of $PLUS$ in the later section, a secure routing.

F. Shared Library

The constructed shared library provides common functions for the routing operator with common routing services and cryptographic operations, for the trust estimator with embedded trust evaluation algorithms, and for the security responser with necessary policies (e.g. table management, threshold checking). The three components are written by calling the shared library.

IV. Evaluation and Result

In this section, we will simulate and analysis the security property and evaluate the performance of our proposed scheme by attacks simulation, cost analysis, and trust value deduction. The simulator is written in C++.

A. Simulation Setup

TABLE III: Simulation Parameters

Simulation area	$140m \times 140m$
Number of nodes	100
Ratio of malicious nodes	2%
Transmission range	$30m$

In the simulated sensor network, nodes are randomly distributed with a uniform density, and can be scheduled randomly be the next traffic source in a data collection application. Herein, we assume a perfect channel, i.e., no packets are lost due to problematic nature of wireless medium; and assume a density network, which just guarantees full connectivity. The main simulation parameters are specified in Table III.

B. Malicious Attack Simulation

In the first set of evaluation, several typical attacks (packets dropping, packets modification, and denial- of-service) are implemented to show the disaster caused by a small quantity of malicious nodes. These nodes can be either non-member brought into the network by the adversaries or member compromised by the adversaries.

1) Attack Patterns:

a) Dropping Attack: In this attack the malicious node acts selfishly. It receives the packets destined to itself, but drops all the packets it supposed to forward after extracting valuable information, such as the sensing data. That is, all the nodes, whose path to base station passing through the malicious nodes, will become unreachable.

b) Modification Attack: In this attack the malicious node adds, alters, or deletes all the packets bypassing itself after extracting valuable information, such as the sensing data. That is, all the nodes whose path to base station passing through the malicious nodes will be

inflected, and the other nodes who are the intermediates between the malicious nodes and destination (sensor node or base station) will be the stupid "accessaries" to continually forward corrupted packets. Compare to the dropping attack, more energy will be consumed and more nodes will be infected.

c) Denial-of-service Attack: Simply speaking, the aim of denial-of-service attack is to against the availability. In Theory, it can be performed in different layers and has various attack misbehavior. Herein, we simulate it in a simple way: malicious node goes into a loop to send frequent unnecessary traffic with random destination to its neighbors, and ask them to forward or process packets, etc. Compare to the above two attacks, more nodes in network will be infected and more energy will be wasted.

2) Simulation Metrics: We will measure performance along two metrics:

a) Number of nodes infected: The number of nodes whose packets sent to the base station is broken due to the attacks executed by malicious node.

b) Additional packets wasted: There are mainly three categories of packets: one for sensing information, one for topology control, and one for trust relation maintenance. Among them, the packets for topology control running in the beginning of network formation period will not be broken; the topology control packets for route maintenance and the packets for trust relation maintenance are basically generated locally for one-hop communication; therefore, the communication packets in the network, which have higher corrupted probability, are sensing data. We consider each hop-to-hop transmission as one packet, and name "wasted" to represent two kinds of packets: the packets discarded in the middle and the forwarded packets which are already corrupted.

3) Simulation Result and Analysis: In order to show more applicable result, we will not fix base station in the central of network, but with uniform random function to let it be in different locations. So does the location of the malicious nodes.

Note that, we run the simulation 100 times and all the analyzed data are averaged from the 100 runs. The results shown are consistent with the different characteristics of the three attack types.

a) Number of nodes infected: Figure 4 shows the infected nodes caused by the three simulated attacks. Among them, the malicious nodes doing DOS attack may infect nearly 30% healthy nodes in the network, while 5% caused by modification attack, 2.5% caused by dropping attack.

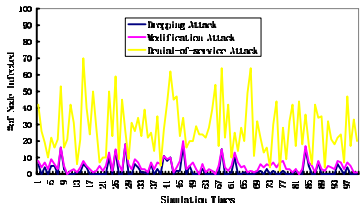

Fig. 4: Node infection caused by attacks

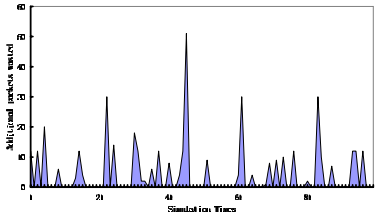

Fig. 5: Additional transmission hops wasted (compare dropping attack to modification attack)

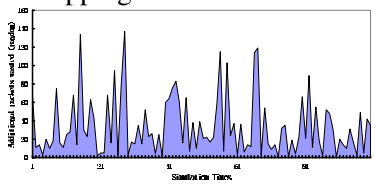

Fig. 6: Additional transmission hops wasted (compare dropping attack to denial-of-service attack)

b) Additional packets wasted: From the Figure 5 and Figure 6, we can easily derive that the wasted transmission hops caused by the modification attack are nearly 3 times more than the dropping attack; while the ones caused by the denial-of-service attack are over 33 times more than the dropping attack.

C. Trust Management Cost Analysis

Trust Level Regulation Table	
Trust Level	Trust Value
1	(0, 0.3)
2	[0.3, 0.5)
3	[0.5, 0.7)
4	[0.7, 1.0)

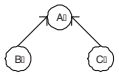

Fig. 7: Network scenario with Regulated Trust

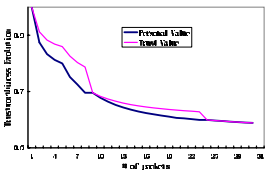

Fig. 8: Trustworthiness Evolution of a dropping node

For providing fault tolerance, [1] and [2] let sensor nodes route packets with multiple paths to any other nodes. This kind of blind redundancy not only consumes the sensor nodes' limited energy and radio bandwidth, but also produces traffic overload and collisions. In $PLUS$, nodes only route packets to trusty entities. The additional communication cost will only be caused for trustworthiness maintenance, which is one hop and short packet transmission; and it is only triggered when trust level of the suspect changes, but not periodically. Since

the trustworthiness computation is mostly the byproduct of packet processing, especially in the in-network processing available network. The additional of computation cost is only caused by few simple statistical computations. The additional memory occupied is almost for storing trust related information in parameter database shown in the section III-A. If the network density is increasing, with the consideration of general sensor device resource constraints, each node can still hold on this amount of records with a good table aging mechanism. And also if a better packet processing mechanism can be provided, the additional memory saving can be achieved.

D. Trust Value Deduction

In this section, we consider a simple network scenario, shown in Figure 7, where node A becomes a malicious node that does dropping attack, node B and C are the downstream nodes towards node A, and they are also mutual neighbors. Through simulation, we show how the node B (act as judge) evolutes the trustworthiness of node A (act as suspect); how the reference coming from node C (act as jury) influences the final trust value computed by node A; and after how many communications, the node B will be regarded having minimal trust; and probably, we can see the communication and computation cost in this specific scenario. The strategy taken here utilizes all of our trust management scheme. And finally, we got the trend-line shown in Figure 8 and we can figure out that there are totally twice trust exchange procedures occurred, and after about 25 communications, the malicious node can be identified.

V. $PLUS\_R$: Parameterized Localized trUst management based Secure Routing for sensor networks

In order to show the benefit of $PLUS$, we design a secure routing protocol $PLUS\_R$, which exploits trustworthy relationship obtained from the above scheme.

We consider a data collection sensor network application where sensor nodes scattered to monitor the environment. The sensing information upwards from a sensor node to the base station along a tree-like network topology. Each node has an unique identifier. And communication is bi-directional. A node is capable of discovering its one-hop neighbors by running some neighborhood discovery protocol. Such protocols are part of most existing routing protocols and therefore can be reused. We also assume that each node is equipped with a local one-hop monitoring mechanism, such as watchdog, to detect misbehaving nodes among its direct neighbors.

A. Protocol Design Issues

Based on the proposed $PLUS$, $PLUS\_R$ need to adapt the parameter database and security responser components.

1) Parameter Database for $PLUS\_R$: The unit parameter $routingMetric$ inside the union parameter $neighborRelation$ should be substituted by the $sumDistrust$ and $routingCost$. And another union parameter $routeInfor$ should be added to describe the parent information of the local node.

```
&routeInfor = (
    parentID
    parentCost
)
```

2) Security Responser for $PLUS\_R$: From routing protocol view, secure routing module of security responser component should has the ability to exploit the calculated trust value to form and maintain a multihop routing topology to support data dissemination. As explained in above, we focus on node to base station communication pattern to form a spanning-tree-like network topology. Route discovery is executed to ascertains the topology of the sensor network and build appropriate routing information just after the network is deployed. And route maintenance is triggered after nodes have certain trust relationship knowledge to form secure topology. Note that, in this section, we will not touch the security mechanisms processed on the route packets.

a) Route discovery: At this stage, network entities have zero-network knowledge. To start route discovery, base station first announces itself. The node that receives this packet will generate its outgoing packet, which includes its identifier, and level identifier (i.e. one). Progressively, each node in network can learn its neighbors. Note that, a node does not immediately send out route packet (herein, *routeDiscovery*). Instead, it starts out a timer immediately after receiving the first *routeDiscovery* to collects all information till expiring timer. And then, the local node selects and records the sender who has minimum level identifier as parent. If necessary, an alternative parent can also be stored. Next, the local node will sends out its *routeDiscovery* with dual functions, route request and rout reply. The packet includes identifier of local node and gradual level identifier. Finally, a tree is formed to be data flows towards the base station where each node has a shortest path towards base station.

b) Route maintenance: Because of the dynamic network status and available network information (i.e. trust value) with time going by, $PLUS\_R$ is ready to run route maintenance to build trustworthy network topology.

The base station initiates the protocol whenever it needs to construct the trustworthy network topology. Same as in route discovery, the base station first announces itself. The node, adjacent neighbor of base station, receives the packet will generate its outgoing packet, which includes its identifier, level identifier (i.e. one), and the distrust value towards base station (usually, zero). Then, the node broadcasts the route packet (herein, *routeMaintenance*). Once received this packet, the receiver (say, node A), checks whether the sender (say, node B) is resident in the black node list, if not, node A does the following operations sequentially:

Algorithm 2 Algorithm for Route Maintenance

update $sumDistrust_B$;
$routingCost_B = neiTrustValue_B + sumDistrust_B$;
update $routingCost_B$;
if $routingCost_B < parentCost$ **then**
  // parent will be node B
  $parentID = B$;
  $parentCost = routingCost_B$;
end if
END

Progressively, each node in network can learn its neighbors. Note that, a node does not immediately send out route packet (herein, *routeMaintenance*). Instead, it starts out a timer immediately after receiving the first *routeMaintenance* to collects all information till expiring timer. Finally, the local node ascertains its parent bypassing which the local node can securely communicate with the base station. If necessary, an alternative parent can also be stored. Next, the local node will sends out its outgoing *routeDiscovery* with dual functions, route request and rout reply. Finally, a tree is formed to be data flows towards the base station where each node has a secure path towards base station.

B. Security performance and communication cost

Once sensor nodes have ability to estimate its neighbor, that is, maintain numerical trust values deduced, they can choose more secure paths to deliver packets to the base station. However, there must be some tradeoff between security performance and communication cost, that is, routing using secure path may cause additional communication cost. Continuing the network scenario, shown as Figure 7, node B and C will reselect their parents instead of node A. We simulate the additional communication cost (i.e. represented by the distance) caused by adopting secure paths but not shortest paths in Figure 9.

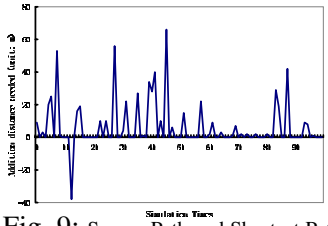

Fig. 9: Secure Path and Shortest Path

From the simulation, we can see the worst case is that 2 hops additional transmission will be caused, however, there are about 5% possibility that node B or C can not reach the base station.

VI. Conclusion and Future Works

In this paper, we present a Parameterized and Localized trUst management Scheme ($PLUS$) for sensor networks security. Compared to the existing works targeted at this field, our scheme is a novel approach from whole system view to well complement current security practices. The highly abstracted parameters give the scheme flexibility to adapt to different operational environment and application domains; the localized trust model is more suitable to the less formal, temporary or short-term trust relationship presented in sensor networks; the derived trustworthiness can be used to conduct efficient security actions and disclose the potential attacks. Lower computational and communication overhead are also achieved in our proposed scheme.

References

[1] Deng, J., Han, R., and Mishra, S., *INSENS: Intrusion-tolerant routing in wireless Sensor Network*, Technical Report CUCS-939-02, University of Colorado, 2002.

[2] Ganesan, D., Govindan, R., Shenker, S., and Estrin, D., *Highly Resilient, Energy Efficient Multipath Routing in Wireless Sensor Networks*, Mobile Computing and Communication, No.2, 2002.

[3] McKnight, Harrison, D., Norman L. Chervany *The Meaning of Trust*, Working Paper, Carlson School of Management, University of Minnesota, 1996.

[4] Perlman, R., *An overview of PKI trust models*, IEEE Network, vol. 13, no. 6, pp. 38-43, 1999.

[5] Perrig, A., Szewczyk, R., Wen, V., Cullar, D., Tygar, J., *Spins: Security protocols for sensor networks*, In Proceedings of the Seventh Annual International Conference on Mobile Computing and Networking, 2001.

[6] Sasha Slijepcevic, Miodrag Potkonjak,Vlasios Tsiatsis, Scott Zimbeck, Mani B. Srivastava, *On communication Security in Wireless Ad-Hoc Sensor Network*, WETICE'02, USA.

[7] Tilak, S., Abu-ghazaleh, N.B., Heinzelman, W., *A Taxonomy of Wireless Micro-Sensor Network Models*, ACM Mobile Computing and Communications Review, 2002.

[8] Yahalom, R., Klein, B., Beth, T., *Trust relationships in secure systems-a distributed authentication perspective*, In Proceedings, IEEE Symposium on Research in Security and Privacy, 1993.

[9] Zhang, Y., Lee, W., *Intrusion detection in wireless Ad-Hoc network*, In Proceedings of The Sixth International Conference on Mobile Computing and Networking, Boston, MA, 2000.

[10] Z.Y. Yao , D. Kim, I. Lee, K. Kim, J. Jang, *A Security Framework with Trust Management for Sensor Networks*, First IEEECREATE-NET Workshop on Security and QOS in Communication Networks (Securecomm 2005), Athens, Greece.