

## Short Paper

---

# Towards Provable Security for Data Transmission Protocols in Sensor Network\*

MI WEN<sup>†</sup>, LING DONG, YAN-FEI ZHENG AND KE-FEI CHEN

<sup>†</sup>*Department of Computer Science and Engineering*

*Shanghai University of Electric Power*

*Shanghai 200090, P. R. China*

*Department of Computer Science and Engineering*

*Shanghai Jiao Tong University*

*Shanghai 200240, P. R. China*

This paper presents a framework to analyze the security of data transmission protocols in wireless sensor network. This framework defines three attack models in terms of the adversary's attacking ability, and provides an ideal model to verify whether a given protocol is secure or not under these three different attack models. Furthermore, we give a formal security definition under different attack models for data transmission protocols. A security checking model is also presented, which describes the detailed steps of how to use the security definition to weigh the security level of a given protocol in the ideal model. A "secure" directed diffusion protocol is analyzed to demonstrate the effectiveness of the security checking model. This is the first time that the notion of provable security is applied in wireless sensor networks.

**Keywords:** provable security, wireless sensor network, data transmission protocol, SDD, attack

## 1. INTRODUCTION

An enormous increase has been seen in the development and use of wireless sensor networks (WSNs) in large scale systems in the past two decades. A basic task in these sensor network systems is to interactively distribute commands to or gather data from a subset of the sensor nodes. Therefore, one of the primary challenges in the designing of these effective sensor network systems is how to guarantee the data transmission protocols, which may run in hostile or unattended environments, robust and secure. Some secure data transmission protocols for sensor networks have been proposed, *e.g.* [5, 8, 14], but their security have been discussed informally and incompletely. Thus, when these protocols are put into applications, they are likely to result in unexpected damages and the protocol need to be amended. Next time, when another error happens, the amending process needs to be repeated. As a result, protocols without any complete an formal proofs of security lead one to question the level of trust in the correctness in such proto-

---

Received March 26, 2007; revised September 13, 2007; accepted November 1, 2007.

Communicated by Tzong-Chen Wu.

\* This work was supported by SRFDP of China under grant No. 20050248043, NSFC of China under grants No. 90704004 and 60703030.

cols. Therefore, a great challenge in WSNs is how to prove the security of data transmission protocols?

To meet this challenge, we present a simulation-based formal framework to prove the security of the data transmission protocols in WSNs. This framework defines three attack models in terms of the adversaries' attacking abilities, and constructs an ideal model to simulate a protocol's operation and analyze its security. Moreover, the framework defines what secure data transmission means under different attack models in terms of their correctness criteria. At last, a "secure" protocol SDD [8] is analyzed to prove if it satisfies those security definitions. In this paper, our attack models and simulation-based approach just focus on provable security of data transmission protocols; they also can be used to address security of other kinds of sensor network protocols. The notion of "provable security" and the simulation approach we used in our framework are following the idea of [1, 10, 11], which has been successfully used to prove the security of cryptographic algorithms [1] and ad hoc routing protocols [10, 11]. "Provable security" in our paper means that a given protocol is formally and completely proved to be secure under a certain level of attacker model. A summarized version of this paper is introduced in [9], which leaves out some details that we contain here. In this paper, we extend the models and security definitions and demonstrate the whole security checking process by analyzing SDD.

The remainder of this paper is organized as follows. Section 2 provides the related works and in section 3, we address the framework of our approach, including attack model and communication model *etc.* We present our formal security definition and security checking process in section 4. In section 5, we demonstrate the usage of our approach by analyzing the security of SDD. Finally, we conclude our paper in section 6.

## 2. RELATED WORK

A large body of literatures dealing with verification of protocols' security has been proposed. And two main computer security protocol analysis techniques are known to people for their wide applications: the theorem proving [6] approach and model checking [2] approach.

Verification systems typically address well-defined properties (the properties may be different for different protocols), such as confidentiality, authentication, integrity, freshness in data transmission protocols, and aim to detect violations of these properties. Theorem proving systems define a set of axioms and relations to prove these properties mathematically. However, the number of axioms and relations grows with the complexity of the protocol. These systems are believed to be even more complex and intractable for data transmission protocols. Moreover these systems work with abstract specifications, and they are not intended for data transmission protocols in WSNs and, thus, it is hard to abstract the wireless nature and various dynamic changes in WSNs.

Model checking systems [2] check the satisfiability of the properties described in their system by manipulating the data variables symbolically, and exploring the control transmission paths explicitly. Consequently, only a finite number of control paths can be checked. If this approach is used to verify the data transmission protocols in WSNs, one of the main obstacles is the undesirability and intractability problems since the control paths includes all the infinite possible sequences of topology changes which result in a state explosion.

Except for the above two techniques, there is another approach to verify protocols which is subjecting them to automatically generated fault oriented tests [3, 4]. The protocols are modeled as finite state machines and rather than checking the entire state space for violations of the correctness criteria, one only checks states reachable from faults. Once incorrect states reachable from faults are identified, a backwards search is performed from the fault to determine if it is reachable. This search is performed for each type of fault and each message of the system and yields a set of tests that lead to error states. The idea of restricting the search to faults helps to mitigate the effect of state space explosion in blind searches of the protocol state space. This paper places emphasis on simulation of the system using the test sequences leading to realistic examples of protocol errors.

### 3. THE FRAMEWORK OF OUR APPROACH

#### 3.1 Network Model

Define the wireless sensor network as an undirected labeled graph  $struc = (V, E, C_v, C_e)$  where  $V$  is the set of vertices, which represents the identities of the participating sensors.  $E$  is the set of edges, which represents the radio links between each pair of sensors. And we assume the radio links are symmetric.  $C_v$  is the cost value of sensors,  $C_e$  is the cost value of the edges. And  $C_v, C_e$  can be used as path selection metric based on application requirements. *i.e.* the assignment  $C_v = 1$  for all  $v \in V$  and  $C_e = 0$  for all  $e \in E$  equals to the hop count metric in WSNs. In addition, we assume that the adversary is static during the operation of the system, which means it can't corrupt more nodes during running of the protocol. The network model has the cryptographic algorithms as components, and we assume all cryptographic algorithms used in the protocol are secure. All of the attacks pointing to the weakness of the cryptographic algorithms themselves are out of the scope of our model.

#### 3.2 Attack Model

In sensor networks, there are two types of adversaries: the passive adversary and the adaptive adversary. The passive adversary has little attacking ability. It only eavesdrops the communication and can't do anything else. Its attacking case can be included in the outsider attack model, so it is not considered as an individual attack model here. The adaptive adversaries are generally be categorized first from their physical abilities as: node-class adversaries and Laptop-class adversaries [13]. The former is further categorized into: Outside adversary and inside adversary (*e.g.* compromised sensor nodes).

The attack models are classified according to the adversaries' attacking abilities. The corresponding attacks in each attack model are stored in the attack tables. Note that, in our model we could not consider the physical attacks which totally destroy nodes permanently because the only method to defend it is to have nodes equipped with tamper resistant hardware [5]. We leave this case in further research. And also we require that the adversary is running in polynomial time.

**(1) The first attack model: Outsider attack**

In this level the attacker can send request to the normal sensor network to ask for joining, if the protocol has not any authentication or authorization mechanisms, the attacker will succeed. This means that an adversary can read and alter those messages that are transmitted by neighboring nodes. If the outsider can't join the network it is also able to hear the message packets and replay messages that are generated by legitimate parties within his broadcast range. The known attacks include eavesdropping, altering, or replaying attacks *etc.*

**(2) The second attack model: Insider attack**

A malicious insider can perform all the attacks that an outsider can. The malicious insiders can attack the network by spoofing or injecting bogus information. Additionally, if the communication messages need to be authenticated by keys, a malicious insider has the keys necessary to generate authentic messages for his own identity. If malicious insiders cooperate and share their keys, each insider may generate any message appearing to originate from any of the compromised nodes. Also the malicious insider can send fake messages to any protocol participant or send observations about the monitored area to a malicious user. So it prevents the networks from functioning as expected. The known attacks include selective forwarding attacks, acknowledgement spoofing, Sybil attack, sinkhole attack *etc.*

**(3) The third attack model: Laptop-class attack**

A laptop-class adversary with a powerful transmitter can actually provide a high quality route by transmitting with enough power to reach the base station in a single hop, or by broadcasting routing or other information with large enough transmission power could convince every node in the network that the adversary is its neighbor. This manner causes a large number of nodes to attempt to send their packets to it, but those nodes sufficiently far away from the adversary would be sending packets into oblivion. A laptop-class adversary also can inject false messages that give incorrect information about the environment to the user. Such messages will consume the scarce energy resources of the nodes or jam the lower layers of communication. The known attacks include sinkhole attacks, wormhole attack, HELLO flood attack and denial of service attack *etc.*

**3.3 Communication Model**

The main idea of the simulation paradigm we used is to construct two models: a real model that describes an instance of the protocol in real environment, and an ideal model that captures the defined specification of the protocol under different attack models. In both models, there is an adversary, whose behavior is not constrained. This allows us to consider any feasible attacks, and makes the approach very general. Later we will prove the ideal model can completely simulate the real model in the statistical sense.

**3.3.1 The real model**

A real model is corresponding to the configuration of  $conf_{real} = (stru = (V, E, C_v, C_e), A)$ , and the system  $SyS_{real,A}$  is described as a real system running in this model.  $A$  repre-

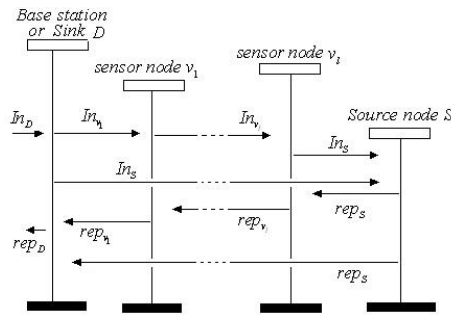


Fig. 1. The communication model of the system.

sents the adversary in the network. We define the whole network communication model for data transmission as Fig. 1. The sinks can either pass messages through intermediate nodes or directly to the source nodes.  $\{D, v_1, \dots, v_i, S\}$  is a set of interactively communicating probabilistic state machines, connected via buffered channels. Each machine can be initialized with some initial states, such as session keys *etc.* The machines operate in a reactive manner, which means that when a machine is activated, it reads the content of its input channels, processes the received data, updates its internal state, and writes some output in its output channels. Each channel is considered as an output channel for the machine before it and an input channel for another machine behind it, and it is also able to determine the target of the message and to which channel it belongs. Machine  $D$  represents the sink. Machine  $v_1, \dots, v_i$  are the intermediate nodes and machine  $S$  is the source nodes.  $In_i$  is the input channel of the machine  $i$ .  $rep_i$  is a response to the request  $In_i$ . Each machine has one more input and output buffered channels, which means our model allows several parallel runs of the protocol.

There are four types of delivery pairs in Fig. 1: sink  $\leftrightarrow$  sensor node, sensor node  $\leftrightarrow$  sensor node, sensor node  $\leftrightarrow$  source node, sink  $\leftrightarrow$  source node. We define the two entities of each pair as Sender and Receiver. We describe the communication between each pair in the real model as Fig. 2. The machine  $T$  is triggered by the information in  $In_T$  from the previous node. Machine  $R$  communicates with machine  $T$  through its channel  $In_R$  and  $rep_R$ . The adversary  $A$  may try to disturb the normal communication, it may tamper the messages in  $In_R$  or  $rep_R$ .

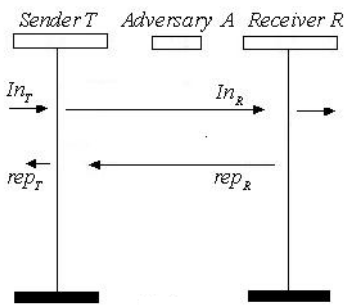


Fig. 2. The pair of sender and receiver in real model.

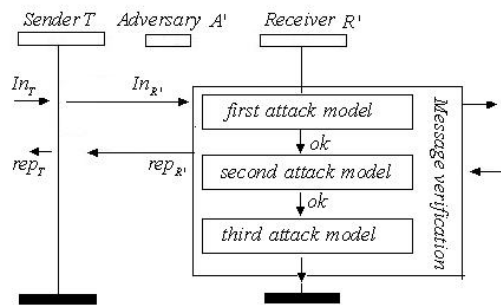


Fig. 3. The pair of sender and receiver in ideal model.

We describe the operation of the machines in more detail as following.

**Machine  $D$**  represents the honest sink, and it also may represent the higher layer protocols. It is triggered by the *interest*  $x$  from its higher manager or its own need.  $D$  can process the *interest*  $x$  in terms of the pre-deployed protocol specifications or application requirements. And it can initiate a data delivery process by sending the processed *interest* to its output channel  $In_{v_i}$ . A response to this interest may be returned via channel  $rep_{v_i}$ .

**Machine  $v_i$**  represents the intermediate node of the network. The operation of  $v_i$  is defined by the data transmission algorithms. When data arrives at its input channel,  $v_i$  is activated and it start to read the contents in its input channel and processes them in the light of initiated instructions, then it writes the results on its output channel. Both channels can buffer messages and states produced in the communication, latter judgments and verifications can refer to that information.

**Machine  $S$**  represents the source nodes whose task is monitoring the environments and diffuse the relevant readings. When new messages arrive, it first processes the contents in the input channel  $In_S$  and determines the task orders. Then it executes in this order and responds via  $rep_S$ .

**Machine  $A$**  represents the adversary. Regarding its communication capabilities,  $A$  is identical to any machine  $T$  or machine  $R$ . However, it may not follow the protocol specification honestly, so it may try to disturb the normal communication between the other machines and tamper the messages in the buffered channels.

In the pair of Sender and Receiver model in Fig. 2, the machine  $T$  and machine  $R$  will be either of the model of machine  $D$ ,  $v_i$  or  $S$ . The whole communication ends when reaches one of its final states. This happens when the time is out or the needed data responding to the interest arrives at the sink. We denote the output of the  $SyS_{real,A}$  as  $Out_{real,A}(x)$  where  $x$  is the initiated *interest*.  $Out_{real,A}(x)$  includes the results buffered in each output channel.

### 3.3.2 The ideal model

The ideal model is corresponding to the configuration  $conf_{ideal} = (stru = (V, E, C_v, C_e), A')$ , and the system  $SyS_{ideal,A'}$  is described as an ideal system running in this model. The whole network communication model for data transmission is same to Fig. 1. The main difference lies in each pair of Sender and Receiver: machine  $R$  is replaced by a new machine  $R'$  and machine  $A$  is replaced by a new machine  $A'$  as described in Fig. 3. Their operation is similar to that in the real model. Therefore we do not detail it here, and we just focus on the new machine  $R'$  and  $A'$ .

**Machine  $R'$**  is initiated with the verification rules (which is set by the protocol specification), errors and attack tables (which contain the possible errors and attacks). When the messages arrive at its input channel  $In_{R'}$  it can do the message verification while processing the contents. Thus, it can detect the system errors. Once an error is found,  $R'$  records

this error in its error-recording tables and the communication continues as if nothing wrong had happened. The details of this message verification process are described in section 4.3. Machine  $A'$  operates similar to machine  $A$  in the real model. We denote the output of the  $Sys_{ideal,A'}$  as  $Out_{ideal,A'}(x)$ . Where  $x$  is the initiated *interest*.  $Out_{ideal,A'}(x)$  includes the results buffered in each output channel.

Similar to the real model, the whole communication ends when it reaches one of its final states.

### 3.4 Unification of the Real Model and the Ideal Model

The goal of this section is to prove that the real model can be completely simulated by the ideal model in the statistical sense.

**Theorem 1** (*statistically security*) Let  $Sys_{real,A}$  and  $Sys_{ideal,A'}$  be systems running in the real model and the ideal model for the same protocol. We say that  $Sys_{ideal,A'}$  is statistically as secure as  $Sys_{real,A}$  for the honest users with the same initial interest  $x$ , if two systems get the same outputs, with the statistical difference of at most  $\varepsilon$ , written as:

$$\varepsilon = \Delta_{sta}(Out_{ideal,A}(x), Out_{ideal,A'}(x))$$

$\varepsilon$  is the negligible probability.

In other words, we demand that for whatever any adversary  $A$  in the real model can do to the real system  $Sys_{real,A}$ , there exists an ideal-model adversary  $A'$  that can achieve the same effects to the ideal system  $Sys_{ideal,A'}$ , with the statistical difference of at most  $\varepsilon$ . The detailed meaning of the “statistical difference” could reference [12].

**Proof:** From the construction of our models and systems, we can see that the steps of the  $Sys_{real,A}$  and the  $Sys_{ideal,A'}$  are exactly the same (for the same interest  $x$ ). If no errors are found in the error-recording tables of the ideal system  $Sys_{ideal,A'}$ , not only the steps, but also the output of the two systems will be the same, namely,  $Out_{ideal,A}(x) = Out_{ideal,A'}(x)$ . So the ideal system is as secure as the real system. On the other hand, if there an error *Malice* is found in the ideal system  $Sys_{ideal,A'}$ , search backward in the simulation paradigm, there must exist an adversary  $A'$ , which launches an attack  $Adv \in A_{tab}$  and leads *Malice*  $\in E_{rtab}$  appear. Apparently, by using this adversary  $A'$  to construct  $A = A'$ , the adversary  $A$  can do the same attack to  $Sys_{real,A}$ . And thus  $Out_{ideal,A}(x) = Out_{ideal,A'}(x)$ . Whereas, if adversary  $A$  in real system can launch an attack  $Adv' \in A_{tab}$  and lead *Malice'*  $\in E_{rtab}$  appear in  $Sys_{real,A}$ .

There may be two possible cases in the ideal system. One is the *Malice'*  $\in E_{rtab}$  will also be arrived by the same attack  $Adv' \in A_{tab}$  in the ideal system, then by following the same specifications,  $Out_{ideal,A}(x) = Out_{ideal,A'}(x)$ . The other one is  $Out_{ideal,A}(x) \neq Out_{ideal,A'}(x)$ . This case is possible only for the following two reasons:

1. Some of the messages during the communication are lost. As a result, the output of some channels in the two systems is different.
2. The attacks aiming at the cryptographic algorithms themselves (*e.g.* find a collision or

pre-image of the hash function) are launched, which are not considered in the ideal model, so the states caused by these attacks can't be found in the channels of the ideal systems.

For reason 1, it happens only due to the nature of wireless communication, some errors appears in the real system can't be repeated in the ideal system (Because wireless communication in the ideal model is assumed to be benign). But for the same interest  $x$ , these failures due to the nature of wireless communication occur only with negligible probability, and it would rather be viewed as a tolerable imperfection of the real system than a flaw of the ideal system.

For reason 2, we has assumed in section 3.1 that the attacks aiming at the cryptographic algorithms themselves are not considered in our model. To say the least, if an attack is launched to break the cryptographic algorithm (*e.g.* hash function  $SHA - 1$ ), it will succeed only with negligible probability within polynomial time. Hence from the sense of statistics,  $Out_{ideal,A}(x) \equiv_{\epsilon} Out_{ideal,A'}(x)$ . Thereby, the ideal system is statistically as secure as the real system. Which shows the  $SyS_{ideal,A'}$  is a simulator for  $SyS_{ideal,A}$ .

**Lemma 2** If the statistical difference between the outputs of the ideal system and the real system is a (small) error  $\epsilon$ . We say the ideal model can completely simulate the real model in the statistical sense.

*Proof:* This lemma can be deduced from Theorem 1.

This property guarantees that once a protocol is proved to be secure in the ideal model, one can ensure that it is statistically secure in the real model under the same attack scenario. Hereafter, before a protocol applying to practice the user only needs to analyze its security in this ideal model, avoiding amending the protocol in iterative process when errors are found in applications.

## 4. CORRECTNESS AND SECURITY DEFINITION

The correctness and security definition here are aiming at the data transmission protocols. Sensor network data transmission protocols can be classified into two categories according to the service policy they follow: Push-based data transmission protocols and pull-based data transmission protocols. This paper just focuses on the pull-based data transmission protocol, which is the most important representative of the data transmission protocols. Directed diffusion protocol [7] is a widely used pull-based data transmission protocol. The advanced one SDD [8] will be analyzed in section 5.

### 4.1 Correctness

We assume the correctness definition of a protocol is given by its applications. Generally, a perfect data transmission protocol should be able to securely transmit data in the presence of any number of attackers. There are four criteria that define the secure data delivery through the network expressed as: *data confidentiality, data authentication, data integrity and data freshness*.

For **data confidentiality**, the network should not leak sensor readings to the adversary. In many applications (e.g. confidential message in battle or martial fields) nodes communicate highly sensitive data. The standard approach for keeping sensitive data secret is to encrypt the data with a secret key that only intended receivers possess, hence achieving confidentiality.

For **data authentication**, the receiver needs to make sure that the data originates from the correct source. Especially for many administrative tasks, authentication is necessary. At the same time, an adversary will inject messages, so the receiver needs to verify that the data used in any decision-making process originates from the authorized source. Often, a message authentication code (MAC) of the communicated data is used to provide the authentication.

For **data integrity**, the receiver wants to ensure that the received data be not altered in transit by an adversary. This goal is always achieved by the data authentication.

For **data freshness**, even if confidentiality and data integrity are assured, we also need to ensure the freshness of each message. That means the data is new, and no old messages have been replayed.

Usually, errors are defined in conjunction with the correctness criteria. In our case, the corresponding errors we address are message leakage, message loss, message modification, and message duplication, which are stored in the possible error table. These errors may manifest themselves in the ways as described in section 3.2.

## 4.2 Security Definition

Now, we are ready to introduce the definition of secure data delivery according to our attack models and correctness criteria.

**Definition 3 (First Attack Model Security, FAMS)** A data transmission protocol is said to be (statistically) secure in the first attack model if, for configuration  $conf_{ideal}$  and any outside node-class adversary  $A'$ , the protocol still satisfies the correctness criteria of its application or none of the protocol errors is found.

In fact, Definition 1 describes the standard requirement we have on protocols in terms of security. However, some protocols may satisfy the following stronger definitions:

**Definition 4 (Second Attack Model Security, SAMS)** A data transmission protocol is said to be (statistically) secure in the second attack model if, for configuration  $conf_{ideal}$  and any inside node-class adversary  $A'$ , the protocol still satisfies the correctness criteria of its application or none of the protocol errors is found.

**Definition 5 (Third Attack Model Security, TAMS)** A data transmission protocol is said to be (statistically) secure in the third attack model if, for configuration  $conf_{ideal}$  and any Laptop-class adversary  $A'$ , the protocol still satisfies the correctness criteria of its application or none of the protocol errors is found.

Synchronizing to the conclusion in section 3.4, the meaning of “statistically secure”

in the above definitions is that once a protocol is proved to be secure in the ideal model, it is statistically secure in the real model. Note that, since the lap-top class adversary has far stronger physical abilities than the sensor nodes, depending only on the soft defense in protocol specification is difficult to ensure the correctness of the protocol. Therefore, it is commonly accepted that the protocol satisfying the security Definition 2 is “acknowledgedly secure”. We define the protocol satisfying the Definition 3 is “perfectly secure”.

### 4.3 Security Checking in the Ideal Model

The security checking in ideal model includes two steps. Firstly, during the data delivery process, the message verification is done at each machine  $R'$  in Fig. 3. This process is beginning from the first attack model, if it fails, an error will be found and recorded in the error-recording table *ExerI*, the verification in this pair of delivery will terminate; if succeeds, goes on checking in the second attack model, similarly if fails, an error will be recorded in the error-recording table *ExerII*, and the verification in this pair of delivery will terminate; if succeeds, goes on checking in the third attack model. Now, if the verification fails also the error will be recorded in the error-recording table *ExerIII*, but if succeeds, it does nothing and goes on the verification in the next pair of delivery. Secondly, when the communication ends, back searches the three error-recording tables to see if there are some errors and determines the security level of the protocols. The Algorithm 1 describes the whole process.

---

#### Algorithm 1 Security checking

Ertab: A table contains the possible error states in data transmission protocol.

AtabI, AtabII, AtabIII: Tables contain possible attacks in three attack models.

ExerI, ExerII, ExerIII: Tables contain the errors found in three attack models.

Adv: is an attack.

Mal: is an error.

**Security checking** // the procedure end when reach one of its terminal state.

{For each (pair of delivery)

{Message sending;

Message verification: // verification for each pair

If exists a  $Adv \in AtabI$  leads to a  $Mal \in Ertab$  is found then

Record the error Mal in ExerI;

Else if exists a  $Adv \in AtabII$  leads to a  $Mal \in Ertab$  is found then

Record the error Mal in ExerII;

Else if exists a  $Adv \in AtabIII$  leads to a  $Mal \in Ertab$  is found then

Record the error Mal in ExerIII; }

#### Determine the security level

{ IF exists a  $Mal \in ExerI$ , Output that " The protocol is insecure!";

Else if exists a  $Mal \in ExerII$ , Output that " The protocol satisfies FAMS !";

Else if exists a  $Mal \in ExerIII$ , Output that" The protocol satisfies SAMS !";

Else Output that " The protocol satisfies TAMS !" ;

}

}

---

## 5. SECURITY LEVEL OF SDD

In this section, we demonstrate the usefulness of our approach. In particular, we simulate the operation of SDD in our ideal model and discover several attacks against the data delivery. And we show that, SDD is insecure in the confidential applications, but it is secure in the first attack model in non-confidential applications.

### 5.1 Operation of SDD

SDD includes four phases, interest propagation, low-rate data propagation and secure routing setup phase, path selection and reinforcement propagation, data propagation.

In the first interest propagation phase, the TESLA protocol [14] is used to ensure that the interest is from sink  $D$ . The one-way chain is generated as  $k_i = F(k_{i+1})$  ( $0 \leq i \leq n - 1$ ), and use reveal. This one-way chain is pre-deployed in the sink and the first element  $k_0$  of this one-way chain is pre-deployed to all sensor nodes. The interest is flooded in the form as:

$$\{H(INTEREST_1) | MAC(k_1, H(INTEREST_1))\}.$$

After time  $d$ , the sink floods another packet  $\{INTEREST_1 | k_1\}$ . The nodes verify if it is from sink by checking if  $F(k_1) = k_0$ . Then it uses  $k_1$  to verify  $H(INTEREST_1)$  by  $MAC(k_1, H(INTEREST_1))$ , and authenticates the  $INTEREST_1$  by  $H(INTEREST_1)$ . Here  $H$  and  $F$  are hash functions, and  $MAC$  represents Message Authentication Code.

In routing setup phase, first, source node  $S$  floods the packet in the following form to let the sink know its first key  $k_S^0$ .

$$\{H(DATA_1) | MAC(k_S^1, H(DATA_1)) | [k_S^0]_{k_S} | S\}$$

Where  $DATA_1$  is the data  $S$  wants to send.  $k_S^0, k_S^1, \dots, k_S^n$  is the source node  $S'$  one way chain used as keys.  $K_S$  is the shared key between node  $S$  and the sink  $D$ . The sink let all other sensor nodes know  $k_S^0$  in the similar form as in the interest propagation. Then, source node  $S$  begins to send exploring data to establish the gradient as following:

$$\{DATA_1 | k_S^1 | nonce_1 | \{S\} | [nonce_1]_{K_S}\}.$$

Where  $\{S\}$  is the node list of the path. Assume this packet will go through node  $E$  and  $L$  as in Fig. 4. The freshness symbol  $nonce_1$  will be encrypted by their shared keys with the sink successively, and the node list of the path will include the nodes this packet passed by. When this packet arrives at the sink, it is in the form of  $\{DATA_1 | k_S^1 | nonce_1 | \{S, E, L\} | [[[nonce_1]_{K_S}]_{K_E}]_{K_L}\}$ . The sink checks the integrity of this packet by decrypting the encrypted symbol  $[[[nonce_1]_{K_S}]_{K_E}]_{K_L}$  using the shared keys in turn. Then it obtains the path  $\{S, E, L\}$ .

In next phase the sink chooses a path probabilistically. If it chooses the path  $\{S, E, L\}$ , it floods  $\{H(REINFORCEMENT_1 | \{S, E, L\}) | MAC(k_3, H(REINFORCEMENT_1 | \{S, E, L\}))\}$ . And later it floods  $\{REINFORCEMENT_1 | \{S, E, L\} | k_3\}$ . The intermediate nodes verify them as in the first phase. Now the path is selected.

At last, when source node  $S$  wants to send data, it sends the following packets to all the intermediate nodes along its selected path.

$$\{H(DATA_2) | MAC(k_S^2, H(DATA_2))\}$$

And later sends  $\{DATA_2 | k_S^2\}$ . The sensor nodes authenticate them similar to the first phase.

## 5.2 Analysis of SDD in Our Ideal Model

From the above we can see that SDD uses the hop count as path metric. Thus, we set the cost assignment as:  $C_v(v_i) = 1$  for all  $v_i \in V$ , and  $C_e(e_i) = 0$  for all  $e_i \in E$ . As described in section 4.3, a data transmission protocol is analyzed in the ideal model to see whether there are errors reachable from the attack models. In order to be compliant with our framework, each phase of the SDD will be considered as the communication between the pair of Sender and Receiver in our ideal model as Fig. 3. The process of our security analysis is performed strictly according to the security checking in Algorithm 1.

First, consider the interest propagation between the sink and sensor nodes. Since the sensors are all pre-deployed a shared key with the sink, any outside adversary can't join them. And with the help of the MAC authentication, any modification will be detected. Also any duplication of the interest can be discarded because of its failure in freshness checking. But the adversary can hear the  $\{INTEREST_1 | k_1\}$  exposed in plaintext later. Thus, the attack 1 may be launched.

**Attack1:** The outsider can eavesdrop the packet  $\{INTEREST_1 | k_1\}$  and knows what the network is processing about. So the data confidentiality is not holding now.

For the confidential applications, this data leakage is the error of the protocol, so they will record this error and end message verification at this phase. But for the non-confidential applications this data leakage can not make against their systems.

And they will go on checking in the second attack model. For any inside adversary in interest propagation, if it pretends to be the sink and sends false interest to other nodes with its own keys  $k_A^1$ , it can't pass the checking, for  $F(k_A^1) \neq k_0$ . So the Sybil attack fails. If it selectively forwards the packets, other nodes will not be influenced because the interest packet is flooded to other nodes. For a laptop-class adversary, it only can do the HELLO flood attack, but it has not the sink's one way keys, so the HELLO flood packets will be discarded for  $F(k_A^1) \neq k_0$ .

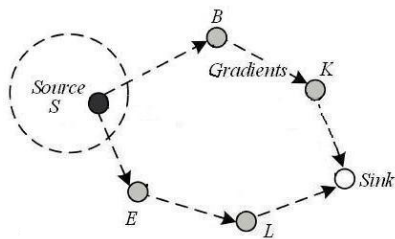


Fig. 4. The routing setup in SDD.

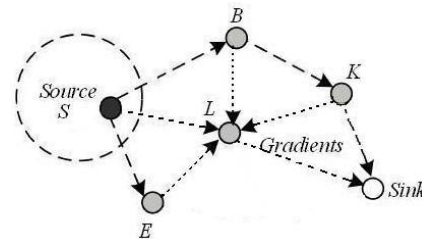


Fig. 5. An attack in the routing setup of SDD.

Second, consider the second phase that the outside adversary can do no harm except the data leakage for the same reasons said in the first phase. For the confidential applications, this data leakage is also the error of the protocol, so they will record this error in their error-recording table and end message verification at this phase. But for the non-confidential applications, they will go on checking. Now, for an inside adversary, it has the legitimate shared key and its own one-way keys, when the routing setup packets pass through it, it can modify the packets as it wants. Thus, the attack 2 may be launched.

**Attack2:** For example, if there is a path  $\{S, E, L\}$  to the sink, but among them,  $E$  and  $L$  are inside adversaries, they could know each other's key and act as a single. Therefore, they can modify the packet  $\{DATA_1 | k_S^1 | nonce_1 | \{S, E, L\} | [[nonce_1]_{k_S}]_{k_E}]_{k_L}\}$  to be  $\{DATA_1 | k_S^1 | nonce_1 | \{S, L\} | [[nonce_1]_{k_S}]_{k_L}\}$  as in Fig. 5, and let the path appear shorter and competitive and tempt the sink to choose this one. Furthermore leads the data in the fourth phase to be sent into oblivion and causes data loss. This is the known Sinkhole attack, and if they can make a tunnel, that will be the wormhole attack. Though the probabilistic path selection can defend them in certain extent, they still will succeed with non-negligible probability. Thus, for the non-confidential applications this attack will lead the error of data loss and it will record this error in its error-recording table and end message verification at this phase.

Next, consider the third phase, the analysis are same as that in the second phase. The reinforcement packet  $\{REINFORCEMENT_1 | \{S, E, L\} | k_3\}$  also will be changed by the inside adversary to be  $\{REINFORCEMENT_1 | \{S, L\} | k_3\}$ . Consequently, the Attack 2 will succeed.

Finally, consider the fourth phase, the analysis are as same as that in the first phase. But we must note that  $\{DATA_2 | k_S^2\}$  in this phase still will be send later in plaintext, this will result in data leakage and the adversary can eavesdrop like Attack 1.

And this error will be recorded in the error-recording table of the confidential applications, and there are no errors for non-confidential applications in this phase.

At the end of the above analysis, we can see that, if SDD is used in confidential applications, it is insecure in confidential applications because it will lead to data leakage from its flaw. If SDD is used in non-confidential applications, though it said in [8] the probabilistic path selection can defend Attack 2 in certain extent, but Attack 2 still will succeed with non-negligible probability. It does not satisfy the proper security definition 4 in our paper. We have to say that it only satisfies FAMS, which means it is statistically secure in the first attack model in non-confidential applications. From the view of our ideal model, to make SDD acknowledged secure in the first application, the encryption mechanism should be changed; to make SDD acknowledged secure in the second application, the best solution is to base the routing setup on the location of the nodes or their distances, in which wormholes and sinkholes are meaningless. Our approach is the first one to check the security of the sensor network protocol by combining the attack model and application requirements.

## 6. CONCLUSION

The main contribution of this paper is that it provides a simulation based ideal

model to prove the security of a protocol. In which the protocol is analyzed according to the defined three attack models and its efficiency is demonstrated by analyzing the “secure” directed diffusion protocol SDD [8].

The significance of this approach is that it can not only help the user to select proper protocols in applications, but also aid the protocol designers in mitigating their effects or reducing the potential pitfalls there. Furthermore, it can even assist the researcher to establish a new model theory for sensor networks in formal method. Another important thing is that it is useful to introduce the sound techniques and notions known in the cryptographic articles and ad hoc networks, and put them to use in the context of sensor network through adaptation. To the best of our knowledge, we are the first who applied the notions of provable security and the simulation paradigm to the sensor networks protocols.

Meanwhile, our current ideal model needs further improvement. We will extend the ideal model by considering the aspects of energy cost and communication overhead in our on-going work. Besides, we will automate the process of the security analysis of the protocols in the ideal model, and relative tools are planned to be developed.

## REFERENCES

1. W. Mao, *Modern Cryptography: Theory and Practice*, Parts V and VI, Prentice Hall PTR, 2004.
2. E. Clarke, O. Grumberg, and D. Long, “Verification tools for finite-state concurrent systems,” *A Decade of Concurrency – Reflections and Perspectives*, LNCS 803, 1994, pp. 124-175.
3. A. Helmy, D. Estrin, and S. Gupta, “Fault-oriented test generation for multicast routing protocol design,” in *Proceedings of the FIP TC6 WG6.1 Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE XI) and Protocol Specification, Testing and Verification (PSTV XVIII)*, Vol. 135, 1998, pp. 93-109.
4. A. Helmy, D. Estrin, and S. Gupta, “Systematic testing of multicast routing protocols: Analysis of forward and backward search techniques,” in *Proceedings of the 9th IEEE International Conference on Computer Communications and Networks*, 2000, pp. 590-597.
5. J. P. Walters, Z. Liang, W. Shi, and V. Chaudhary. “Wireless sensor network security: A survey,” <http://www.cs.wayne.edu/~weisong/papers/walters05-wsn-security-survey.pdf>.
6. I. Cervesato, N. Durgin, P. D. Lincoln, J. C. Mitchell, and A. Scedrov, “A meta-notation for protocol analysis,” in *IEEE Proceedings of the 12th Computer Security Foundations Workshop*, 1999, pp. 55-71.
7. C. Intanagonwivat, R. Govindan, D. Estrin, *et al.*, “Directed diffusion for wireless sensor networking,” *IEEE/ACM Transactions on Networking*, Vol. 11, 2003, pp. 2-16.
8. X. Wang, L. Yang, and K. Chen, “SDD: Secure distributed diffusion protocol for sensor networks,” in *Proceedings of the 1st European Workshop on Security in Ad Hoc and Sensor Networks*, LNCS 3313, 2005, pp. 205-214.
9. M. Wen, L. Dong, Y. F. Zheng, and K. F. Chen, “A framework for proving the security of data transmission protocols in sensor network,” in *Proceedings of Proceed-*

- ings of Pacific Asia Workshop on Intelligence and Security Informatics*, LNCS 4430, 2007, pp. 288-294.
10. G. Acs, L. Buttyan, and I. Vajda, "Provable security of on-demand distance vector routing in wireless ad hoc networks," in *Proceedings of the 2nd European Workshop on Security and Privacy in Ad Hoc and Sensor Networks*, LNCS 3813, 2005, pp. 113-127.
  11. L. Buttyan and I. Vajda, "Towards provable security for ad hoc routing protocols," in *Proceedings of the ACM Workshop on Security in Ad Hoc and Sensor Networks*, 2004, pp. 94-105.
  12. D. Hofheinz and D. Unruh, "On the notion of statistical security in simulatability definitions," in *Proceedings of the 8th International Conference on Information Security*, LNCS 3650, 2005, pp. 118-133.
  13. C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: attacks and countermeasures," *Ad Hoc Network Journal, Special Issue on Sensor Network Applications and Protocols*, 2003, pp. 293-315.
  14. A. Perrig, R. Canetti, D. Song, and J. D. Tygar, "Efficient and secure source authentication for multicast," in *Proceedings of the IEEE Symposium on Network and Distributed System Security*, 2001, pp. 35-46.

**Mi Wen (温蜜)** received the B.S. degree from Northeast Normal University, received her M.S. degree in Computer Software and Theory from University of Electronic Science and Technology of China, in April 2005, and received her Ph.D. degree in the Department of Computer Science and Engineering, from Shanghai Jiao Tong University of China, in June 2008. Now, she is a lecturer of Shanghai University of Electric Power. Her research interests are in wireless networking, especially on wireless sensor networks with an emphasis on routing and network security.

**Ling Dong (董玲)** is currently a Ph.D. candidate in the Department of Computer Science and Engineering, Shanghai Jiao Tong University. His research interests include provable security and secure protocol design, etc.

**Yan-Fei Zheng (郑燕飞)** is currently a lecture and Ph.D. candidate in the Department of Computer Science and Engineering, Shanghai Jiao Tong University. Her research interests include security and key management in wireless sensor networks.

**Ke-Fei Chen (陈克非)** received his Ph.D. degree in Justus Liebig University Gies-sen, Germany, 1994. His main research areas are classical and modern cryptography, theory and technology of network security, *etc.* Since 1996, he came to Shanghai Jiao Tong University and became the Professor at the Department of Computer Science and Engineering. Up to now (1996-2008), he has published more than 100 academic papers on cryptology.