Formal Analysis of a Ranked Neighbour MANET Protocol Suite

Shamim H. Ripon, Syed Fahin Ahmed, AfrozaYasmin, Yeaminar Rashid, and K. M. Imtiaz-Ud-Din

***Abstract*—Formal verification plays an important role in development and application of safety critical systems. Formalized verification techniques to analyze the security and the safety properties of communication protocols increase and confirm the protocol confidence the advancement of mobile and wireless communication technologies in recent years introduced various adaptive protocols to adapt the need for secured communications. Security is a crucial success factor for any communication protocols, especially in mobile environment due to its ad hoc behavior. SPIN is a powerful model checker that verifies the correctness of distributed communication models in a rigorous and automated fashion. This paper presents a SPIN based formal verification approach of a security adaptive protocol suite. The protocol suite includes a neighbor discovery mechanism and routing protocol. The protocol suite is encoded into SPIN and is exhaustively checked for various temporal properties ensuring the applicability of the protocol suite in real-life applications.**

***Index Terms*—Ranked neighbor, SPIN, model checking, loop freedom.**

I. Introduction

Mobile Ad Hoc Networks (MANETs) provide many challenges to the research community. Secure routing is one of them. There are many proposals in the literature that claim to provide certain levels of security in routing functionality [1], [2]. However, there are few works that formally analyze the security of these proposals.

When security is crucial for a protocol, the design should be proven to be secure. The most reliable way to do this is to deploy formal methods. Formal methods help us either to prove or to refute the correctness of a design. In this work, we use model checking approach. Specifically, we use PROMELA (PROcessMEta-LAnguage) [3] as the specification and modeling language and SPIN (Simple PROMELA Interpreter) [4] as the model checker. Our primary aim is to demonstrate design flaws that lead to violations of security requirements using model checking. Model checkers are good at finding design errors and they provide error traces (i.e., counter-examples). In our case, an error trace demonstrates a successful attack.

Many wireless networking mechanisms, notably routing, require that wireless nodes be aware of their neighborhood. This means that the nodes must know which other nodes they can communicate with directly. The procedure used to acquire this knowledge is called neighbor discovery [1]. In mobile wireless networks, the neighbor relationships change dynamically, which makes neighbor discovery an important mechanism. Neighbor discovery can be achieved through simple protocols, where a node that wants to determine who its neighbors and broadcasts a neighbor discovery request, and every node that receives this request responds with a neighbor discovery reply. Receiving a reply means that the requesting node and the responding node can hear each other's transmission, and can communicate with each other directly, and hence are neighbors. The neighbor discovery protocol is sometimes called "Hello protocol", and the request and the reply are called "Hello messages" [5].

An adversary can try to thwart the successful execution of the neighbor discovery protocol, for instance, by jamming the communication between two nodes, or by providing a node with false information regarding another node, which is not a direct neighbor in reality, but leading on to make the requesting node believe that the other node is indeed a direct neighbor. In this way, the adversary achieves that two nodes, which otherwise could communicate directly, cannot establish a neighbor relationship, or a relationship with faulty information [5]-[7]. Blocking the links between many pairs of nodes in this manner can have serious consequences to the connectivity of the network, and on the upper layer protocols, such as routing and transmission.

Numerous approaches have been proposed to analyze the security and routing properties of ad hoc protocols. These techniques include visual inspection, network simulation, analytical proofs, simulatability models and formal methods [8]. Formal verification is one of the most reliable ways to rigorously check the desired properties of a protocol by modeling and analyzing it mathematically.

Model checking is an automated technique that, given a finite-state model of a system and a logical property, systemically checks whether this property holds for that model. Within appropriate constrains, model checker can perform an exhaustive state-space search on a software design or implementation and alert the implementing organization to potential design deficiencies by producing a counter example.

The rest of the paper is organized as follows. Section II gives a brief review of AODV protocol and describes the extended version of the AODV protocol that we are going to model check in this paper. After giving a short overview of SPIN and PROMELA in Section III we outline how the security protocol suite is encoded into SPIN for model checking purpose. We also describe the mechanism to model the required security properties to be verified. The following section shows how the proposed security properties are verified in SPIN. Finally, we conclude the paper by

Manuscript received March 20, 2014; revised May 30, 2014.

S. H. Ripon is with the East West University, Dhaka, Bangladesh (e-mail: dshr@ewubd.edu).

S. F. Ahmed, A. Yasmin, Y. Rashid, and K.M. Imtiaz-Ud-Din are with the Department of Computer Science and Engineering, East West University, Dhaka, Bangladesh.

DOI: 10.7763/IJFCC.2014.V3.320

summarizing our results and outlining our future plan.

II. Security Adaptive Protocol Suite: Overview

A MANET (Mobile Ad-hoc Network), is a spontaneous network of computers where nodes can join or leave at any time and are free to move around as they desire. As there is no centralized infrastructure the participating nodes need to function both as end nodes and routers. Due to limited transmission range, packets that are destined to nodes outside of transmission range have to be routed over some intermediate node(s). If one node has a path to another node, packets are expected to be routed there correctly.

AODV is an on-demand or reactive protocol designed for use in MANET. Routes from one node to other nodes are discovered on demand. AODV supports unicast, broadcast, and multicast. The route decisions are made using distance vectors. In distance vector routing, each router contains distances, measures in hops or in some other forms, to all available routers in a routing table. Since this is an on demand distance vector, routers only maintain distances of only those destinations that they need to contact or relay information to. Due to the adaptive nature of this approach, the protocol is conveniently deployed in highly mobile network environment.

During route discovery, when a node requires a path to destination, it broadcasts a Route Request message to its neighbours. Each node receiving the message creates a reverse route to the source node. The destination node or intermediate node that has up-to-date route information sends back Route Reply message. Each node receiving the route reply message creates a forward route to the destination. Thus each node remembers only the next hop required to reach any of the hosts.

Security Adaptive Protocol Suite [9] is an extension of AODV protocol. The suite is a combination of a neighbor discovery mechanism and a routing protocol. In the neighbor discovery phase, the neighbor nodes are ranked. The ranks are determined by calculating their distances. Ranks considered the trustworthiness of the nodes. With the ranked information collected expressing the trust of the neighbors, the routing protocol proceeds. When a demand is made from a source to a destination, the source node first judges the security requirement of the application and then based on the ranking information of the neighbor nodes the packet is routed to the destination.

Estimated physical distance between nodes is used to rank the neighbors. If a node is physically further away than that in routing table, a wormhole is considered to be present in the network. The maximum allowed transmission distance is added to the packets to restrict the routes.

Based on the calculated distance values the sender decides the trust values by ranking the neighbors based on predefined range. Four (4) types of ranks are used in the protocol suite. Table 1 shows the ranks, where,$d'$is the calculated distance of a neighbor and T is the maximum allowed distance.

Each neighbor node is assigned a rank. Whenever a route request is made from a source to a destination, the minimum security level (MSL) is checked with the assigned ranked values and transmit the message only through the trusted nodes. With the information of ranking, it possible to distinguish the wormholes present in the network.

TABLE I: Rank Assignment

Distance Estimation	Rank (R)
$d' \leq T/4$	4
$T/4 < d' \leq T/2$	3
$T/2 < d' \leq 3T/4$	2
$3T/4 < d' \leq T$	1
$T > d'$	0

When a route is requested from a source node, A, to a destination node, B, in AODV, a route request is broadcasted. In the case of security protocol suite, the basic principle will be same, the only difference lies in the fact that, before a route request is broadcasted, the security level requirement has to be defined, named as the Minimum Security Level (MSL). This will be done on a predefined scaling basis, related to the rank values for the nodes. Thus, with a security requirement level, the node now broadcasts the route request only to the nodes with a minimum level of trust. This specific operation is being possible to be implemented, as because AODV is able to unicast, as well as multicast. The neighbors, upon receipt of the request, will react the same way, as it is done in the traditional AODV, except for the slight change in the way it rebroadcast.

III. The SPIN Model Checker

SPIN is an automata-based temporal logic model checker. It accepts design specification written in the verification language PROMELA (Process Meta Language). In PROMELA, a system is modelled as a composition of asynchronous processes that interact with buffered or un-buffered message channels. Since there is no notion of time or clock, models with real-time aspects are very hard, if not impossible, to express in PROMELA. The language is especially designed to describe systems such as asynchronous communication protocols.

Correctness properties of a specification are specified in several ways in PROMELA. *Assertions* and *Never*claims are the most frequently used constructs for this purpose. An assertion has similar semantics as C Language. When the asserted expression becomes false, the assertion fails, and SPIN gives *assertion violation* error.

Never claims are used to specify the finite or infinite behavior that should never happen during the execution of asystem. When a property to be satisfied bythe system is specified, it is formalized in a logic formula and produce aneverclaim that corresponds to the negation of this formula.SPIN then tries to find violations for this never claim. If itfinds one, this means there is a case that the opposite of the specified property can be occurred in the system, which means the propertycannot be satisfied by the system.Anever claim can be written by hand or can be translatedfrom a linear temporal logic (LTL) formula. SPIN alsoincludes a timeline property editor that helps usersvisuallyspecify properties that are otherwise hard to formalize.

SPIN has two modes of operation: simulation and verification. In simulation mode, it runs the model and helps users both to get an impression on how their model behaves and to debug their model. In verification mode, on the other

hand, SPIN analyses the model against the given properties considering all possible executions by performing an exhaustive search on the state space. It can also perform partial search on the state space, which is quite useful in case of very large models or insufficient computational resources. If SPIN finds a violation, it produces an error trace. Using this error trace, a user can run a simulation of the execution that leads to the violation.

Our primary aim in this work is to verify the properties of the security adaptive protocol suite. We employ SPIN to check the protocol model against some properties that we formallyspecify as never claims in PROMELA and list any flaws, if any, as violations.

IV. Protocol Implementation Using SPIN

Model checking in SPIN is often bounded by the amount of physical memory available to the computer. To alleviate this problem it is required to reduce the complexity of the model. A simplified version of security adaptive protocol is being used in our experiment to avoid unnecessary details irrelevant to our verification. When modeling an ad hoc network protocol, apart from the usual consideration of limiting state space, it is required to pay attention to the way of modeling broadcast, connectivity as well as the dynamics of topology.

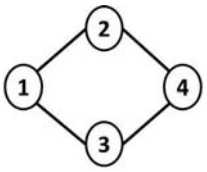
Broadcast is heavily used in most ad hoc networking protocol and it can be modeled by unicasting to all nodes with whom the sending nodes has connectivity. A HELLO message is used to maintain contact with its neighbors and also to contact new neighbors. Hello messages indicate the presence of a node.

Hello message are frequently sent by each node via channel. HELLO message is supposed to have only one piece of information: HELLO massage, source id number which identifies the node from where the hello message is coming from

PROMELA doesn't provide any time features except a timeout function. Timeout keyword is a modeling feature that provides an escape from a hang state which does not correspond to the real timer definition. So in this tools use time as variable to maintain the clock time.

Link update is required to maintain the ranks of the neighbors. Because a link update travels through the network, it represents the most up to date network topology information. When a route is requested from a source node, A, to a destination node, B, in AODV, a route request is broadcasted. For the security protocol suite, the basic principle will be same, the only difference lies in the fact that, before a route request is broadcasted, the security level requirement has to be defined, being called here as the Minimum Security Level(MSL). Here node A checks the trusted neighbors to send message. So this node checks the security level of all its neighbors. The source node will be notified of the fact that the route request that has been sent is not returning a path with the defined MSL. In this scenario, the source will now define a new MSL, by decrementing the MSL value, and will rebroadcast the Route Request to a new set of nodes.

A routing table is maintained for each node. Whenever a packet is to be transmitted from one node to another.



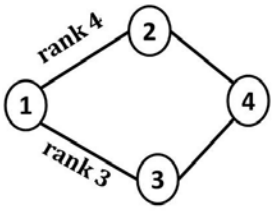
Routing table for node 1

Destination	Next-hop 1	Next-hop 2	Hop
4	2	4	2
4	3	4	2
3	3	-	1
2	2	-	1

Routing table for node 3

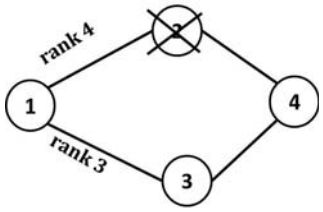
Destination	Next-hop 1	Next-hop 2	Hop
2	4	2	2
2	1	1	2
4	4	-	1
1	1	-	1

Fig. 1. Routing information for node 1 and node 3.



Destination	Next-hop 1	Next-hop 2	MSL Level
4	2	4	Rank 4

Fig. 2. Route table with node rank (MSL).



Destination	Next-hop 1	Next-hop 2	MSL Level
4	3	4	Rank 3

Fig. 3. Route update when MSL changes.

Routing table is consulted along with the rank information of each node and keeping the MSL value. The process of message broadcasting reference to MSL will help the route request reach the destination with trusted neighbor. Fig. 1 illustrates the route table of two nodes.

Suppose Node 1 in Fig 2 is broadcasting a message destined to Node 4. The MSL level is kept at rank 4. In case a route is not able to be established with the initial MSL, with which the *Route Request* was broadcasted, an *Error* packet will be generated. Then source will define new minimum MSL, by decrementing the MSL value and will rebroadcast the *Route Request* to new set of nodes (Fig 3).

A. Property Specification

Among the various properties related to any SAODV protocol, we are interested in two properties crucial to the earlier mentioned protocol suite. The first one is the loop freedom and another is the maintenance of correct rank of the neighbours based on their distances. It is also need to be

confirmed that packets are always transmitted only through designated ranked nodes. In case of a violation, SPIN will produce an error trace that gives the details of the attack.

Loop Freedom: A routing protocol is loop-free if its operation does not allow the formation of cycles on a route between two nodes. If there are *n* number of nodes in the network, then loop freedom ensures that the length of a route can bbe at most *n-1*. The loop freedom property can be stated as follows:

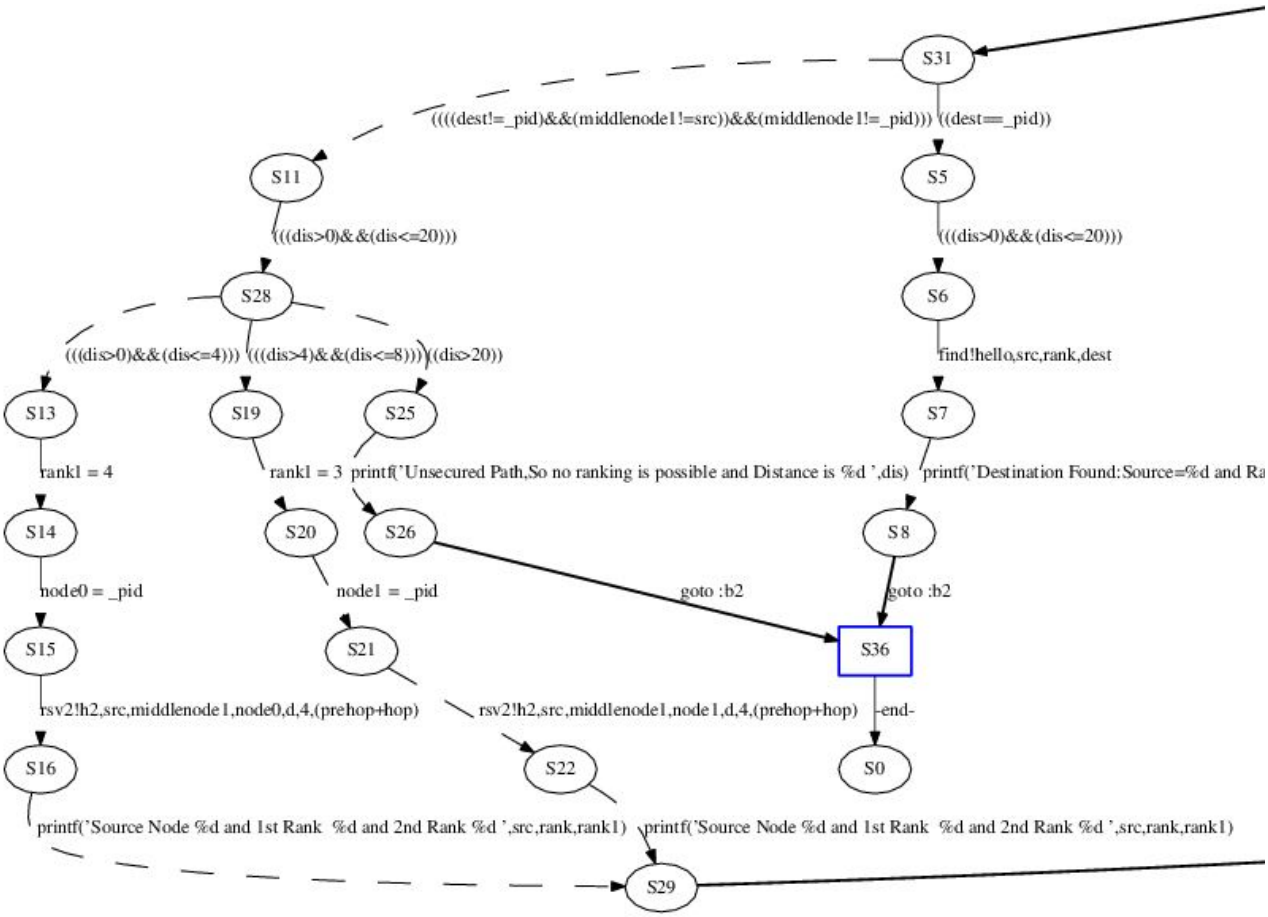


Fig. 4. Automata view of a node (partial).

"It is always true that if node S has a route entry for a destination node D then node D must be at most *n-1* hops away from node O, where n is the number of nodes in the network."

Loop freedom is independent of the network topology. The property can be specified as an assertion within the model or as a never claim or as a LTL formula. The corresponding LTL formula for the loop freedom property can be specified as follows:

$$(p \rightarrow q)$$

where p stands for the "if" part and q stands for the "then" part of the property.

Correctness of rank Information: The requirement for the maintenance of correct rank information property can be informally stated as follows:

"It is always true that if there is a route from an originator node O to a destination node D, the length of this route known to node O is actually the secured path, i.e., follows the rank order from O to D". As the ranks are calculated based on the estimated distance between nodes, this property also ensures that the route is always the shortest path from source to destination. The corresponding LTL formula for this property is as follows:

$$(p \rightarrow q)$$

where, p stands for the proposition "There is a route from O to D" and q stands for the proposition "The rank of this route known to O is actually the highest rank from O to D". This formula is converted to a never claim in PROMELA, which represents the behavior that should never happen. Therefore, we negate our original formula and convert it to a corresponding never claim by using the internal converter supplied with SPIN. The corresponding never claim for the correctness of distance property is as follows:

```
never { /* !([] (p->q)) */
T0_init:
if
:: (! ((q)) && (p))
->gotoaccept_all
:: (1)
->goto T0_init
fi;
accept_all:
skip
}
```

B. SPIN Verification

Because the verification is bounded by the amount of

physical memory available on the computer, we need to reduce the system complexity. Complexity reduction is the most important part of the verification. Due tomobility, the state space increases exponentially with the complexity of the protocol model. Therefore, we modeled a five node ad-hoc network. Five nodes represent enough different network configurations that a verification done on it provides a great probability of good behavior for larger networks. We verify properties of protocol and not performance. A basic model of the protocol suite has been made whichwas simplified by reducing the code and the number of properties included in the model until a good verification result was obtained.

The XSPIN graphic interface is used in order to make simulations. An important number of simulations have been made before finding a model which suits the requirements. A part of XSPIN simulation of the protocol suite is depicted in Fig. 4 and Fig. 5 shows the automata view of a node produced in SPIN. Due to limited space, only a portion of the automata is illustrated.

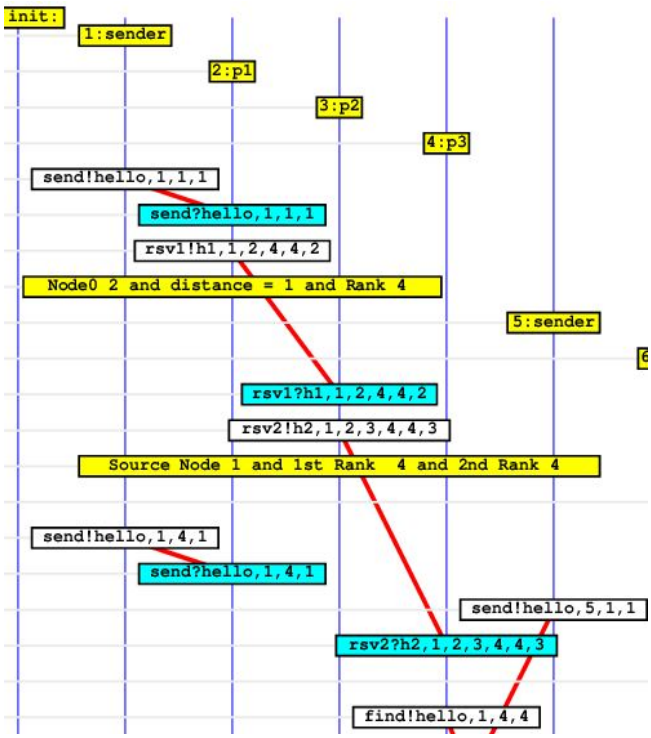


Fig. 5. Simulation of protocol suite (partial).

At the initial stage of verification, we did some simple verification, such as confirmation of message broadcast. If channels are defined properly for each node, when a node broadcast a Hello message, other nodes will receive the message.

An assert statement in SPIN is similar to skip in the sense that it is always executable and has no other effect on the state of the system than to change the local control state of the process that executes it. A very desirable side effect of the execution of this statement is, however, that it can trap violations of simple safety properties during verification and simulation runs with Spin.

The assert statement takes any valid Promela expression as its argument. The expression is evaluated each time the statement is executed. If the expression evaluates to false (or, equivalently, to the integer value zero), an assertion violation is reported.

Assertions can be placed between any two statements of a program and the model checker will evaluate the assertions as part of its search of the state space. Assertions are statements consisting of the keyword assert followed by an expression. When an assert statement is executed during a simulation, the expression is evaluated. If it is true, execution proceeds normally to the next statement; if it is false, the program terminates with an error message.

In SPIN, we use assertion for verification of properties. The property is that, source wants to send message to destination only on its rank 4 neighbors, not the other ranked neighbors. So assert expression as follows:

$$\text{assert}(!(d0!=0 \;\&\&\; d0<t/4);$$

A simulation, instead of a verification, will not necessarily prove that a safety property expressed with an assert statement is valid, because it will check its validity on just a randomly chosen execution.Note that, placing a system invariant assertion inside a loop cannot guarantee that a simulation would check the assertion at every step. Recall that the fact that a statement can be executed at every step does not guarantee that it also will be executed in that way.

Earlier, we mentioned that loop freedom is crucial to successful verification of a protocol. Loop freedom can be checked by counting number of hops traversed by a packet from sender to destination. We define a LTL formula to check this property which is verified by the verification option in SPIN. The automata view of the LTL is as shown in Fig.6.

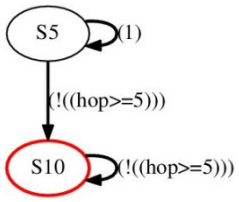


Fig. 6. Automata view of LTL.

V. Conclusions

This paper has shown the use of model checking tool SPIN to formally verify two important properties of a ranked neighbor discovery protocol suite. Model checking of the properties confirms the claimed strength of the protocol. While performing the experiments several improvements have been made to the original description of the protocol.

While encoding the protocol, additional constraints were added to the node description to avoid loop while searching for neighbours. Loops to already discovered route are also avoided carefully. SPIN verification confirms that while routing packets, the protocol ensured that the selected route consists of the highest ranked neighbours and the packets are traversed in the shortest path.

In comparison with other formal verification technique, theorem proving in particular, model checking is task is automatic. Besides, verification result is generated with a fraction of second. For any error or property violation, counter example is also generated to trace the error location.

We are at the early stage of analyzing various properties of the protocol. Our future plan includes model checking other

properties, especially, timing behavior, mobility as well as multiple attacks. We are also interested to employ theorem proving technique by using HOL, PVS to prove the securities in parallel with model checking.

REFERENCES

[1] H. Y. Chun and A. Perrig, "A survey of secure wireless ad hoc routing," *Security and Privacy, IEEE* , vol. 2, no. 3, pp. 28, May-June 2004.

[2] P. G. Argyroudis and D. O'Mahony, "Secure routing for mobile ad hoc networks," *IEEE Communications Surveys*, vol. 7, no. 3, pp. 2–21, 2005.

[3] G. J. Holzmann, *Design and Validation of Computer Protocols*, Prentice Hall, November 1990.

[4] G. J. Holzmann, *Spin Model Checker, The: Primer and Reference Manual*, Addison Wesley, September 2003

[5] L. Buttyán and J-P. Hubaux, *Security and cooperation in Wireless Networks*, July 27, 2007.

[6] C. E. Perkins and E. M. Royer, "Ad-hoc On-Demand Distance Vector Routing," in *Proc. 2nd IEEE Workshop on Mobile Computing Systems and Applications*, 1997, pp. 90-100.

[7] J. Katoen, *Concepts, Algorithms and Tools for Model Checking*, IMMD, 1999.

[8] D. B. Johnson and D. A. Maltz , *Dynamic Source Routing in Ad Hoc Wireless Networks in Mobile Computing*, Edited by Tomasz Imielinski and Hank Korth, Kluwer Academic Publishers, 1996

[9] R. H. Khan, K. M. I. Din, A. A. Faruq, A. R. M. Kamal, and A. Mottalib, "Security adaptive protocol suite: ranked neighbor discovery (RND) and security adaptive AODV (SA-AODV)," in *Proc. International Conference on Electrical and Computer Engineering (ICECE 2008)*, pp. 588 593, 2008.

Shamim Ripon is an associate professor in the Department of Computer Science and Engineering, East West University, Dhaka, Bangladesh. He leads software engineering and formal method research group. Previously, he was a research associate in the Department of Computing Science, University of York, UK and Research Fellow in the Department of Computing Science, University of Glasgow, UK. He also served as a Lecturer in Khulna University, Bangladesh. He is a member of IAENG, IEB, and a senior member of IACSIT. Ripon holds a B.Sc. in computer science and engineering from Khulna University, MSc in computer science from National University of Singapore and PhD in computer science from University of Southampton, UK. His research interests focus on the formal method, requirement engineering, software product line, semantic web, natural language processing. His current research examines the formal representation and verification of knowledge based requirement specification.

Syed Fahin Ahmed completed his B.Sc in computer science and engineering from East West University, Dhaka, Bangladesh in 2013. He is a member of Software Engineering and Formal Method Research Group, East West University.

Fahin is working as a graduate teaching assistant in East West University. He is interested in formal analysis of wireless security protocols, software engineering,

AfrozaYasmin is a BSc final year student in the Department of Computer Science and Engineering, East West University, Dhaka, Bangladesh. she is involved in the software engineering and formal method research group.

Afroza is currently involved in the model checking manet security, protocols by using SPIN model checker. She is also interested in web services and intelligent agent.

Yeaminar Rashid is a BSc final year student in the Department of Computer Science and Engineering, East West University, Dhaka, Bangladesh where he is involved in the software engineering and formal method research group.

Yeaminar is currently involved in the model checking SAODV security protocols by using SPIN model checker. She is also interested in software engineering and knowledge management system.

K. M. Imtiaz-Ud-Din is a senior lecturer in the Department of Computer Science and Engineering, East West University, Dhaka, Bangladesh. He is a member of network and security research group. Previously, he served as a lecturer in Islamic University of Technology.

Imtiaz holds a B.Sc. in computer science and engineering from Islamic University of Technology (IUT) and MScin Information Technology, Security & Mobile Computing from KTH, Sweden, NTNU, and Norway. He research interest includes cloud computing, pervasive systems and self adaptive systems.