A Topology-Hiding Secure On-Demand Routing Protocol for Wireless Ad Hoc Network

Niroj Kumar Pani
Department of
Computer Science Engineering
& Application
IGIT, Sarang, India

Bikram Keshari Rath
Department of
Computer Science
Utkal University
Bhubaneswar, India

Sarojananda Mishra
Department of
Computer Science Engineering
& Application
IGIT, Sarang, India

ABSTRACT
Secure routing in wireless ad hoc network has been an active area of research and in recent years, a number of secure routing protocols have been introduced. These solutions may be classified as proactive, reactive, and hybrid based on the routing information update mechanism deployed. Studies reveal that the reactive (on-demand) ones often outperform the others due to their ability to adjust the amount of network overhead created to track the mobility in the network. However, the existing secure on-demand routing protocols have also some limitations. In this paper, the weakness of existing popular secure reactive routing protocols is analyzed on the ground of topology exposure. Then a new topology-hiding secure on-demand routing protocol, called TSOR is proposed based on timestamp approach and asymmetric cryptography. Security analysis of TSOR shows that it efficiently defeats all possible threats imposed by external or internal adversaries. Simulation results demonstrate that our protocol has a better capability of finding reliable and shortest routes in the presence of malicious nodes at the cost of low routing overhead.

Keywords
Ad-hoc networks, Security attacks, Secure routing.

1. INTRODUCTION
Routing is a fundamental networking function for each and every node in a wireless ad hoc network (also known as, mobile ad hoc networks - MANETs), which makes it lucrative for attackers aiming at disabling the operation of the whole network.

Attacks against MANET routing [1-3] can be classified as passive or active. The passive attack does not disrupt the normal functioning of the network instead, the attacker snoops the data exchanged in the network in order to extract valuable information. Here the requirement of confidentiality gets violated. Detection of passive attack is very difficult since the operation of the network itself is not affected. Examples of passive attacks include eavesdropping, traffic analysis, and traffic monitoring. On the other hand, an active attack attempts to alter or destroy the data being exchanged in the network and hence disrupts its normal operation. Active attacks may violate the security requirements of integrity, authenticity, availability, and non-repudiation. Active attacks can be internal or external. External attacks are carried out by nodes that do not form a part of the network whereas internal attacks are from compromised nodes from within the network. Since the attacker already belongs to the network, internal attacks are more severe and hard to detect than external attacks. Examples of active attacks include impersonation attack, modification attack, fabrication attack, routing loop formation, packet redirection, Sybil attack, DoS attack, rushing attack, black hole attack, wormhole attack, and gray hole attack.

In recent years, a number of secure routing protocols have been proposed. A survey of the protocols is given in [4-7]. These solutions may be classified as proactive, reactive, and hybrid based on the underlying routing information update mechanism employed. It has been studied that the reactive (on-demand) ones often outperform the others because of their low network overhead.

In this paper, we make two contributions to the area of secure routing in MANET. First, we thoroughly analyze the exploits of existing popular secure on-demand routing protocols on the ground of their topology exposing/hiding nature. Second, we detail the design of a new topology-hiding secure on-demand ad hoc network routing protocol, called TSOR that relies on timestamp approach and highly secure asymmetric cryptography to provide the required level of security. In comparison to the related previous works, TSOR is more secure and efficient. We analyze its robustness against possible security attacks imposed by external and internal compromised nodes. We also conduct intensive performance evaluation through simulation, which shows that TSOR has a better capability of finding reliable and shortest routes in the adversarial scenario.

The rest of the paper is organized as follows. Section 2 discusses the security exploits of the existing popular secure on-demand routing protocols. In Section 3, we detail the design of our proposed protocol TSOR. Security analysis of the protocol is given in Section 4. The performance evaluation is conducted in Section 5. Section 6 concludes this paper.

2. RELATED WORKS
A number of secure on-demand routing protocols have been proposed in near decades that are either completely new stand-alone protocols or in some cases incorporation of security mechanism into existing non-secure ones like DSR and AODV.

In order to analyze the existing secure on-demand routing protocols in a structured way, they are classified them into two groups based on the packet content, particularly whether or not the packets (route requests and replies) carry certain information such as recorded routes or hop counts that expose the network topology. This criterion for classification is felt to be important, because the topology related information, without any firm method of confidentiality, may become vulnerable and can be directly used by malicious nodes to launch an attack. On the other hand, the absence of such information, as we shall see, may lead to a sub-optimal path or the formation of routing loops. In our classification, the first group consists of those protocols in which the packets

carries topology exposing information such as recorded routes or hop counts. Representative protocols include SRP [8], Ariadne [9], SDSR [10], and SAODV [11]. The other group includes protocols such as ARAN [12] and SADSR [14] in which the packets have no such information. The weaknesses of each group of protocols are examined in this section.

2.1 Topology-Exposing Protocols

SRP, Ariadne, and SDSR are the extensions of DSR. They adopt the concept of source routing. All these protocols carry the whole route record from source to destination in packet headers. SRP and Ariadne use the one-way hash chain to provide end-to-end integrity. They assume that a secret key is shared between the source and the destination. On the other hand, SDSR provides end-to-end integrity by applying digital signature, which also ensures hop-to-hop authentication. For key management SDSR assumes the presence of trusted certification authorities (CAs). Putting the whole route information in the packets incorporates two major problems. First, with only integrity and authentication checks and without any firm method to ensure confidentiality of the packet content, as these protocols do, the route information present in the packets exposes the network topology. This is a serious issue [15] because the malicious nodes can deduce a part or the whole network topology from the captured route record present in the packets and may launch many possible attacks, such as black hole attack, wormhole attack, and Sybil attack. Second, putting the complete route information in the packets makes the packet size larger. For a small ad hoc network it may not be a problem, but for a relatively large network where the source and destination are separated by a number of intermediate nodes, putting the complete route record within the packets increases the packet processing time and hence consumes more battery power of the nodes. The problem becomes particularly harsh in the case of SDSR because it uses digital signature (an asymmetric key primitive) that has a high computational overhead. Ariadne, Context, and SRP have less packet processing time in comparison to SDSR as they apply one-way hash chain (a symmetric key technique) to provide packet integrity. However, a hash chain doesn't provide hop-to-hop authentication. So an intermediate malicious node can easily modify or fabricate the packets. Moreover, the shared key itself may be compromised.

Unlike SRP, Ariadne, and SDSR, SAODV doesn't carry route information in the packets. However, the packets (route requests and route replies) in SAODV include the hop count information which is used by the protocol as the routing metric for determining the optimal route. In order to prevent the manipulation of the hop count, SAODV uses hash chains. The basic idea is that the originator of a route request or a route reply sets the hop count field to 0. The max hop count field is set to the value of time to live field. The node then generates a random number and sets the hash field equal to it, and applies the hash function specified by the corresponding field max hop count times to the random number. The result is stored in the top hash field. The integrity of the max hop count and the top hash fields are preserved by digital signature. As the packet traverse the network, every intermediate node rehashes the value of the hash field (max hop count minus hop count times) to verify whether the result matches the value of the top hash field. Then, before forwarding the packet, the node increases the value of the hop count field by one and updates the hash field by rehashing its value once. The rationale behind proposing this hash chaining is that an intermediate node can verify the value of the hop count field but cannot compute the preceding hash values because of the one-way property of the hash function. This ensures that an adversary cannot decrease the hop count, thereby, cannot make a route appearing shorter than it really is. However, the later claim made by the authors doesn't hold, because although an intermediate malicious node cannot decrease the hop count field, it may pass on the packet without increasing the value of the hop count field and without updating the value of the hash field. In this way, a compromised node can advertise the shortest route through it and launch a black hole attack.

2.2 Topology-Hiding Protocols

Two representative protocols ARAN and SADSR are examined. Both the protocols uses digital signature for packet integrity and hop-to-hop authentication. For public key distribution and management, the presence of trusted certification authorities (CAs) is assumed. A node before entering the network obtains an off-line certificate from a CA. The certificate binds the IP address of the node with its public key. A simple ad hoc network given in Fig. 1 where S is the source node and D is the destination node. ARAN works as follows. The source S initiates a route discovery process by broadcasting a signed route request packet RREQ: <RequestPacketId, D.IP, S.Certificate, S.Nonce, TimeStamp>S.PubKey. The nonce and timestamp are used to prevent routing loops. The node N1, on receiving the RREQ, verifies the signature of S. If it is found to be valid, N1 signs the RREQ with its own secret key, then rebroadcasts the signed RREQ and its own certificate (<RREQ> N1.PubKey, N1.Certificate). After receiving N1's packet, N2 verifies and removes N1's signature and certificate, signs the RREQ, appends its own certificate and rebroadcasts the message. In this way, the RREQ reaches node D through N1-N2-N3-N4. The destination D replies to the first RREQ that it receives from a source and a given nonce. D creates a signed route reply packet RREP: <ReplyPacketId, S.IP, D.Certificate, S.Nonce, TimeStamp>D.PubKey and unicasts it to the source along the reverse path that the RREQ traverses. The intermediate nodes N4, N3, N2, and N1 repeat the same technique to verify the authenticity of the RREP that they have followed while forwarding the RREQ. In addition to the RREQ and RREP, the nodes in ARAN use signed error packets (ERR) to notify the source of an inactive or broken link. For example, in Fig.1 when N2 receives an RREQ from S and finds that the link N2-N3 is broken, it creates a signed error packet ERR: <ErrorPacketId, S.IP, D.IP, N2.Certificate, N2.Nonce, TimeStamp>N2.PubKey. The ERR packet is unicasted to the source S along the reverse path and authenticated by each intermediate node.

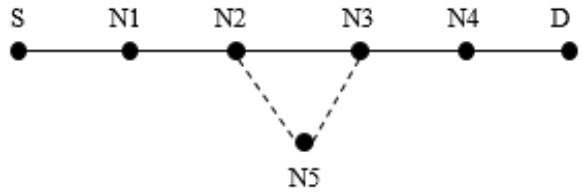

Fig. 1. A simple ad hoc network with seven nodes

It can be noticed that in ARAN none of the packets (RREQ, RREP, or ERR) contain a route record or a hop count information. Therefore, ARAN is not prone to any attack that the topology-exposing protocols like SRP, Ariadne, SDSR, and SAODV do. However, ARAN is vulnerable to some other serious attacks as given next.

- In Fig. 1 lets assume that N3 is a malicious node. On receiving the RREP from D, N3 should unicast the RREP

to N2 because the path of RREQ is S-N1-N2-N3-N4-D. However, N3 can redirect the RREQ to N5. In such a situation N5 could not be aware of N3's malicious intention since N3's signature and certificate are valid. Finally, when the RREP reaches the source S through the path N4-N3-N5-N2-N1, the integrity check in the source becomes valid since N3 does not modify the RREP. This limitation of ARAN has been pointed out by the authors in [10]. It falsifies the claim of authors of [12] and [13] that ARAN guarantees a secure optimal path.

- Let us reconsider the ad hoc network given in Fig. 1, but this time assume that the nodes N2, and N3 are malicious. These nodes don't show any illegal behavior while forwarding an RREQ, for example, an RREQ initiated by S may reach D through the path S-N1-N2-N3-N4-D. However, while unicasting the RREP the malicious nodes may show their intention by forming a routing loop, for example when N2 receives an RREP from N3, instead of unicasting the RREP to N1, N2 sends it back to N3, which again forwards the RREP to N2. The nodes N2 and N3 may continue this looping once or two times before the node N2 forwards the RREP to N1. The intention of the malicious nodes here is to make a routing delay. To the best of our knowledge, this weakness of ARAN has not been published yet.
- A malicious node, for example, N2, or N3 may even launch a gray hole attack by dropping some or all RREP/ERR packets. It may lead to a DoS attack.

The same attacks happen to SADSR. So the discussion is not given here.

It can be observed that, the redirection and routing loop attacks are not possible on ARAN if either a route record or a hop count information is present in the RREQ or RREP, because, in that case, the source S compares the route record or the hop count information present in the RREP with that of the RREQ and the mismatch is caught.

3. TSOR

This section presents the proposed protocol Topology-hiding Secure On-demand Routing (TSOR). There are three objectives in designing TSOR: (1) topology information is completely hidden, so that the malicious nodes cannot deduce the network topology; (2) even with the necessity of hiding topology, TSOR can detect any possible attack, including those against the topology-hiding secure routing protocols that were discussed in the previous section; (3) Once a threat is detected TSOR can exclude the unreliable route (a route that contains one or more nodes exhibiting malicious behavior) before transmitting packets. To achieve these goals, TSOR employs the following mechanisms. In order to ensure that the network topology is kept hidden, none of the routing packets in TSOR carry either a route record or a hop count information. Thus, no node can deduce network topology by capturing the packets. For detection of attacks and exclusion of malicious routes, TSOR uses two techniques. First, end-to-end integrity and hop-to-hop authentication of packets are enforced. This is done on the basis of the work of ARAN. Second, a stringent hop-to-hop verification of packet traversal time is imposed. A combination of these determines the reliability of a path.

TSOR operates in two phases: route request phase and route reply phase. Before these phases are discussed in detail, we present our assumptions and the data structures used in TSOR.

3.1 Assumptions and Data Structures

TSOR uses digital signature for end-to-end integrity and hop-to-hop authentication. We assume the presence of trusted Certification Authorities (CAs) for an authentic public key set up. A node before entering the network must obtain a certificate from a CA. The certificate binds the IP address of the node with its public key, which is used by other nodes to verify the packet signed by the node. In addition to this, it is assumed that all the nodes are tightly clock synchronized with a permitted error (clock difference) of Δ. The value of the parameter Δ must be on the order of a few milliseconds, and must be known to all the nodes in the network. This type of time synchronization has been already been used in [16]. Tight time synchronization is necessary for accurate verification of packet traversal time.

In terms of data structures, a node in TSOR maintains two tables. One is the route discovery table (RDT) which is used to store information about the route request packets (RREQ) processed by the node during the route request phase. An entry in the RDT corresponds to one RREQ processed by the node. The RDT contains the following fields:

- Src: IP address of the source from which the RREQ is originated.
- Dst: IP address of the final destination of the RREQ.
- Src.T_D: Departure time of the RREQ from the source. This time is set by the source node and is present in the RREQ itself.
- T_A: Arrival time of the RREQ at the current node.
- T_D: Departure time of the RREQ from the current node.
- PrevHop: IP address of the previous hop from which the RREQ is received.

The other is the routing table (RT) which is created by a source node S only after the route reply phase is complete and the source finds a route to the desired destination. The RT contains three fields.

- Dst: IP address of the destination node.
- NxtHop: IP address of the next hop through which to reach the destination.
- T_T: Traversal time of the RREQ.

3.2 Working

Now the operation of TSOR is discussed in detail. As already mentioned, TSOR works in two phases: route request phase and route reply phase.

Route Request Phase: The route request phase is initiated by a source S when it needs a route to a destination D but fails to find a route in its routing table (there is no entry for D in the routing table of S). In this case, the source S creates a route request packet (RREQ) which contains the following fields:

- REQ: Request packet type identifier.
- S: IP address of the source S.
- D: IP address of the destination D.
- S.T_D: Departure time of the RREQ from S.
- S.Cert: Certificate of S.

The tuple <S, D, Src.T_D> uniquely identifies an RREQ. S then signs the RREQ using its private key (S.PriKey) and broadcasts the packet. However, before broadcasting, S inserts a new entry in its route discovery table (RDT) for this RREQ and also sets a timer. If the source S doesn't receive a route

reply packet (RREP) from the destination D before the timer expires, it reinitiates a route request phase. A route request phase is also reinitiated by the source if it receives an RREP within the timer expires but finds that the security of the RREP has been compromised (the RREP has reached S through a malicious path). The source S can make up to η route request attempts (η is application dependent). The value of the timer is dynamic and normally based on the round-trip-time.

Every intermediate node on receiving the RREQ, first verify the previous nodes' signature using the previous nodes' public keys found in their respective certificates. The intermediate nodes then examine the uniqueness of the RREQ by checking the tuple <S, D, $S.T_D$> in their RDTs. If the RREQ is found to be authentic and unique (the first one), the intermediate nodes remove the previous nodes' signature (if the intermediate nodes are not the immediate neighbors of the source S), sign the RREQ, append their own certificates, insert an entry in their RDT for this RREQ, and rebroadcast it. Instead, the RREQ is rejected.

Eventually, the RREQ reaches the destination D. The first action taken by D upon receiving the RREQ is to examine the authenticity of the packet by verifying the signatures of its previous hop as well as the signature of the source S. If one of the signatures is found invalid, the RREQ is rejected. On the other hand, if both the signatures are valid, the destination D checks the tuple <S, D, $S.T_D$> in its RDT to determine whether this is the first RREQ for this route request attempt (recall that, a source can make up to three route request attempts). There may be three cases.

- Case 1: The tuple <S, D, $S.T_D$> completely matches with an earlier entry in the RDT of D. It means that the current RREQ is a duplicate copy of an already processed RREQ of the recent route request attempts made by the source. So the current RREQ is rejected.
- Case 2: Only the tuple <S, D> matches with an earlier entry in the RDT, but $S.T_D$ is different. It means that the current RREQ is the first route request packet of another (2nd and onwards) route request attempt made by the source. An earlier attempt has already been made by the source for which an RREP has been sent to the destination D. However, the RREP has been rejected by the source because the packet has reached the source S through a malicious route. In such case, it becomes necessary for the destination D to determine whether the current RREQ has reached D through the same malicious route through which the earlier RREP (the rejected one) has reached the source S. To know this, D compares the traversal time (T_T = RREQ's arrival time - $S.T_D$) of the current RREQ with the traversal time of that entry in the RDT whose <Src, Dst> matches the tuple <S, D> of the current RREQ. If the traversal time of that entry (T_A - $Src.T_D$) is found to be within the range $T_T \pm \Delta$, it implies that the RREQ has reached the destination D through the same malicious path. Hence, the current RREQ is rejected. Instead, the destination accepts the RREQ, deletes the earlier entry in the RDT whose <Src, Dst> matches the tuple <S, D> of the current RREQ, inserts a new entry in the RDT for this RREQ and initiates a route reply phase.
- Case 3: The tuple <S, D, $S.T_D$> doesn't match with an earlier entry in the RDT. It means that the current RREQ is the first route request packet of the first route request attempt made by the source. Therefore, the destination D accepts this RREQ, inserts a new entry in its RDT for this RREQ and initiates a route reply phase.

Route Reply Phase: The route reply phase is initiated by the destination D in response to its acceptance of an RREQ sent by the source S. In this phase the destination D creates a route reply packet (RREP) which has the following fields:

- REP: Reply packet type identifier.
- D: IP address of the RREP's source.
- S: IP address of the RREP's destination.
- $S.T_D$: Departure time of the RREQ from the source S against which this RREP is generated
- T_A: Arrival time of the RREQ at D against which this RREP is generated.
- $D.T_D$: Departure time of the RREP from D.
- D.Cert: Certificate of D.

D then signs the RREP using its private key (D.PriKey) and unicasts the packet through the reverse route (a reverse route is automatically set because every intermediate node before forwarding an RREQ during the route request phase set the PrevHop field in their route discovery table).

When an intermediate node along the reverse path receives the RREP, it verifies the signature of the node from which it receives the RREP. If it is found invalid the packet is rejected. Instead, the intermediate node compares the RREP's traversal time from the destination to the node (RREP's arrival time - $D.T_D$) with the traversal time of the RREQ from the node to the destination against which this RREP is generated (T_A - T_D). $D.T_D$ and T_D are present in the RREP itself, whereas T_A can be obtained from the route discovery table (RDT) of the intermediate node by comparing the RREP's tuple <D, S, $S.T_D$> with the RDT's entries. If there is a difference of more than Δ it means that, either a packet redirection attack or a routing loop attack has occurred on the RREP. Hence, the intermediate node drops the RREP. This approach has the advantage that the other nodes along the reverse path from the current node up to and including the source node don't have to unnecessarily process a compromised RREP. On the other hand, if the traversal times of the RREP and the corresponding RREQ have a difference of less than or equal to Δ, the intermediate node considers the RREP as a valid one. So, the intermediate node removes the signature of the node from which it has received the RREP (if the intermediate node is not the immediate neighbor of D), signs the RREP, appends its own certificate, and unicasts the RREP to the next node in the reverse path. However, before unicasting the RREP, the intermediate node deletes the corresponding RREQ entry from its RDT.

Finally, the RREP reaches the source node S. The source S, on receiving the RREP, first verifies the signatures of the node from which it receives the RREP as well as the signature of the destination D. The source S, then compares the traversal time of the RREP with that of the corresponding RREQ in the similar manner as the intermediate nodes do. If everything is fine the source accepts the RREP, deletes the respective RREQ entry in its RDT, and makes a new entry in its RT (Dst = D, NxtHop = The node form which S has received the RREP). Otherwise, the RREP is rejected and the source S reinitiates an RREQ phase.

The detailed protocol is given in Algorithm 1.

Algorithm 1: **/*** The procedures TsorSource (S, D), TsorDestination (S, D), and TsorNode (S, D, X) are respectively executed by the source S, the destination D, and an arbitrary intermediate node X. ***/**

(1) TsorSource (S, D)

attempt = 0 ; /* Initialize the number of attempts*/
timer = t; /* Initialize the timer */
RouteRequest (S, D); /* Procedure call */
attempt = attempt +1;
start timer; /* Decrement the timer with each clock pulse*/
while (attempt <= η) **do**
if (timer == 0) **then** /*Reinitiate route request*/
timer = t;
RouteRequest (S, D);
attempt = attempt +1; start timer;
else
if (type of received packet == RREP) **then**
if (verifyNodeSign (RREP) and verifyDstSign (RREP)) **then**
/*Verify the signature of the node from which the RREP is received as well as the signature of the destination*/
T_T = RREP's arrival time - RREP[D.T_D];
RDT.R[T_D] = T_D of that entry in RDT whose <Src, Dst, Src.T_D> is same as RREP's <D, S, Src.T_D >;
t = RREP[T_A] - RDT.R[T_D];
if ((T_T - Δ) <= t <= (T_T + Δ)) **then**
/* Accept the RREP */
delete (RDT entry having RDT[Src] == S and RDT[Dst] == D);
/* Insert an entry in the RT */
RT[Dst] = D;
RT[NxtHop] = IP address of the node from which the RREP is received;
RT[T_T] = T_T;
end if
else
reject (RREP);
end if
end if
end if
end while

(2) RouteRequest (S, D)

/* Create a RREQ */
S.Sign = <REQ, S, D, S.T_D, S.Cert>S.PriKey;
RREQ = <<REQ, S, D, S.T_D, S.Cert>, S.Sign>;
delete (RDT entry having RDT[Src] == S and RDT[Dst] == D);
/* Clear RDT and insert an entry in the RDT for this RREQ */
RDT[Src] = S; RDT[Dst] = D;
RDT[Src.T_D] = CurrentTime; RDT[T_A] = NULL;
RDT[T_D] = CurrentTime; RDT[PrvHop] = NULL;
broadcast (RREQ); /* Broadcast the RREQ */

(3) TsorDestination (S, D)

if (verifyNodeSign (RREQ) and verifySrcSign (RREQ)) **then** /*Verify the signature of the node from which the RREQ is received as well as signature of source. */
if (there exists an entry in RTD such that the tuple <RREQ[S], RREQ[D], RREQ[S.T_D]> == the tuple <RDT[Src], RTD[Dst], TD[Src.T_D]>) **then**
reject (RREQ); /* Duplicate RREQ */
end if
if (there exists an entry in RTD such that the tuple <RREQ[S], RREQ[D]> == the tuple <RDT[Src], RTD[Dst]>) **then** /* 2nd and onwards route request attempt */
T_T = RREQ's arrival time - RREQ[S.T_D];
RDT.R[T_A] = T_A of that entry in RDT whose <Src, Dst> is same as RREQ's <S, D>;
RDT.R[Src.T_D] = Src.T_D of that entry in RDT whose <Src, Dst> is same as RREQ's <S, D>;
t = RDT.R[T_A] - RDT.R[Src.T_D];
if ((T_T - Δ) <= t <= (T_T + Δ)) **then**
/* RREQ reached through a malicious path */
reject (RREQ);
else /* Accept the RREQ */
delete (RDT entry having RDT[Src] == S and RDT[Dst] == D);
/*Insert an entry in RDT for the RREQ*/
RDT[Src] = S; RDT[Dst] = D;
RDT[Src.T_D] = RREQ[S.T_D];
RDT[T_A] = RREQ's arrival time;
RDT[T_D] = CurrentTime;
RDT[PrvHop] = The node from which the RREQ is received;
RouteReply (RDT[Dst], RDT[Src], DT[PrvHop]);
end if
end if
if (for all entries in RTD the tuple <RREQ[S], RREQ[D], RREQ[S.T_D]> != the tuple <RDT[Src], RTD[Dst], RTD[Src.T_D]>) **then**
/* 1st route request attempt, Accept the RREQ */
/* Insert an entry in RDT for this RREQ */
RDT[Src] = S; RDT[Dst] = D;
RDT[Src.T_D] = RREQ[S.T_D];
RDT[T_A] = RREQ's arrival time;
RDT[T_D] = CurrentTime;
RDT[PrvHop] = The node from which the RREQ is received;
RouteReply (D, S, RDT[PrvHop]);
end if
else /* Invalid signatures */
reject (RREQ);
end if

(4) RouteReply (D, S, RDT[PrvHop])

/* Create a RREP */
D.Sign = <REP, D, S, S.T_D, T_A, D.T_D, D.Cert>D.PriKey;
RREP = <<REP, D, S, S.T_D, T_A, D.T_D, D.Cert>, D.Sign>;
unicast (RDT[PrvHop], RREP); /*Unicast RREP*/

(5) TsorNode (S, D, X)

if (type of received packet == RREQ) **then**
if (verifyNodeSign (RREQ) and verifyUnique (RREQ)) **then** /*Verify the signature of the node from which the RREQ is received, and also the uniqueness of the RREQ by comparing the RREQ's tuple <S, D, S.T_D> with the RDT's tuple <Src, Dst, Src.T_D >*/
removePrevSign (RREQ);
/* Remove previous node's sign if it is not the source */
X.Sign = <RREQ>X.PriKey;
RREQ = <<RREQ>, X.Sign, X.Cert>;
/* Insert an entry in RDT for this RREQ */
RDT[Src] = S; RDT[Dst] = D;
RDT[Src.T_D] = RDT[S.T_D];
RDT[T_A] = Arrival time of the RREQ;
RDT[T_D] = CurrentTime;
RDT[PrvHop] = the node from which the RREQ is received;
broadcast (RREQ); /* Broadcast RREQ */
else
reject (RREQ);
end if
end if
if (type of received packet == RREP) **then**
if (verifyNodeSign (RREP)) **then**
T_T = RREP's arrival time - RREP[D.T_D];
RDT.R[T_D] = T_D of that entry in RDT whose <Src, Dst, Src.T_D> is same as RREP's <D, S, Src.T_D >;
t = RREP[T_A] - RDT.R[T_D];
if ((T_T - Δ) <= t <= (T_T + Δ)) **then**
removePrevSign (RREQ);
X.Sign = <RREQ>X.PriKey;
RREQ = <<RREQ>, X.Sign, X.Cert>;
delete (RDT entry having RDT[Src] == S and RDT[Dst] == D);
unicast (RDT[PrvHop], RREP);
end if
else

```
            reject (RREP);
        end if
    end if
```

4. SECURITY ANALYSIS

The combination of techniques deployed in TSOR makes it resilient against all possible damages incurred by malicious nodes either internal or external to the network. We first discuss the robustness of TSOR in resisting the following major attacks that could be effectively launched against a secure on-demand routing protocol from a random position in the network.

- *Impersonation attack:* In TSOR every node is required to authenticate its neighbor during both route request and route reply phases. Only those packets are accepted that are signed with a certified key issued by a CA. Therefore, it is impossible for an external or an internal compromised node to impersonate any other node.
- *Modification attack:* Since all packets are signed by the initiating nodes (RREQ is signed by the source and RREP is signed by the destination), any alterations in transit are immediately detected and the altered packet is subsequently discarded.
- *Fabrication attack:* Fabrication attack is prevented, because in TSOR no intermediate node is allowed to generate a routing packet.
- *Black hole attack:* In TSOR, the intermediate nodes are not allowed to confirm or disconfirm the availability or reliability of a route either directly by sending a packet to the source or indirectly by modifying the packet content (such as modifying the hop count field as in the case of SAODV). Only the source and the destination has the capacity to choose a reliable path. So the chance of a black hole attack is eliminated.
- *Wormhole attack:* TSOR can resist wormhole attack because (1) it is topology hiding and hence it is impossible for attackers to choose central positions in the network to launch the attack and (2) neither it uses hop count as a routing metric nor it allows an intermediate node to modify the packet content.
- *Sybil attack:* TSOR does not include identity nor topology information in the routing packets, and thus it is impossible for malicious nodes to obtain the identity information of other nodes. Therefore, TSOR is resistant to Sybil attack.

Next, we examine TSOR against the packet redirection attack, the routing loop attack, and the packet drop attack as presented in Section 2, which the existing topology-hiding protocols fail to defy. We consider the ad hoc network given in Fig. 1 for our discussion.

- *Packet redirection attack:* Let us assume that the RREQ which is accepted by the destination D has reached D through the path S-N1-N2-N3-N4-D. Suppose node N3 is malicious. During the route reply phase, when D unicasts the RREP through the reverse path, node N3 launches a packet redirection attack by unicasting the RREP to N5 instead of unicasting it to N2. This attack is immediately caught by the node N2 when it receives the packet from N5, because when N2 compares the traversal times of this RREP with that of the RREQ, it finds a difference of more than the permitted error Δ. If the packet transmission delay between any two nodes is taken to be Ω and node's packet processing time to be μ, then the traversal time of the RREQ from N2 to D is $3\Omega+2\mu$, whereas the traversal time of RREP from D to N2 through N5 is $4\Omega+3\mu$. So the difference is $\Omega+\mu$. In such case, N2 rejects the RREP. When the source S doesn't receive an RREP within the time out it reinitiates the route request phase. However, this time, the destination D accepts the first RREQ which reaches D trough a different path. Therefore, the malicious path (that contains N3) is automatically excluded.
- *Routing loop attack:* Let us consider that the nodes N2, and N3 are malicious. They launch a routing loop attack in the route reply phase as follows. When N2 receives the RREP from N3, it sends the packet back to N3, which again forwards it to N2. This type of attack is also easily detected in TSOR, because when N1 receives the RREP from N2 (after few loops) and compares the traversal times of this RREP with that of the RREQ, it finds a difference of more than the permitted error Δ (each loop takes $3\Omega+3\mu$ time) and the RREP is subsequently discarded by N2.
- *Packet drop attack:* The packet drop attack is prevented because when an RREP is dropped by some intermediate node, the source doesn't receive an RREP within the timer expires. Hence, it reinitiates the route request phase. But this time, the destination doesn't accept the RREQ which has traversed through the same malicious path that the earlier RREQ does. So, automatically the malicious node is excluded.

5. PERFORMANCE EVALUATION

5.1 Simulation Methodology

The performance TSOR was evaluated using the NS2 network simulator [17]. TSOR is compared with ARAN with and without the packet redirection and routing loop attacks. As discussed in Section 2.2, though ARAN is a topology hiding protocol, it is prone to these attacks. Both protocols were simulated using a 512 bit key and 16-byte digital signature. We used 802.11 MAC layer and CBR traffic over UDP. Node movement was simulated according to the random waypoint mobility model [18]. Ten CBR sessions were simulated in each run, with random source and destination pairs. Each session generated 1000 packets. A random delay between 0 to 10 ms was introduced before a packet is broadcasted in order to minimize collisions. Other parameters are listed in Table 1.

Following performance matrices are evaluated for a non-adversarial scenario when there were no malicious nodes.

- *Average packet delivery fraction:* It is the average ratio of packets received by the destination to the packets generated by the source. It evaluates the ability of the protocol to discover routes.
- *Average routing load in packets:* It is the average ratio of control packets overhead to data packets.
- *Average path length:* It is the average length of the paths discovered by the protocol. It was calculated by averaging the number of hops taken by each received packet.
- *Average route acquisition latency:* It is the average delay between the sending of an RREQ by a source to a destination and the receipt of the first corresponding RREP by the source.

In order to examine how TSOR performs against ARAN in an adversarial scenario when the malicious nodes were executing packet redirection and routing loop attacks, the matrices average path length and average route acquisition latency are evaluated, as discussed above, against each attack. These two matrices were considered important under adversarial setting

because they respectively indicate the extent of path elongation and delay in ARAN under the packet redirection and routing loop attacks. Longer routes result in greater routing overhead and longer delays.

Table 1. Simulation parameters.

Parameters	Values
Simulation area	1000 m × 1000 m
Channel capacity	2 Mb/s
Number of mobile nodes	50
Node transmission range	250 m
Node movement speed	0, 1, 5, 10 m/s
Pause time between movements	30 s
Packet generation rate	4 packets/s
Packet size	512 bytes
Packet processing delay (including signature generation and verification times)	8 ms (5.5 ms for signature generation, 0.5 ms for signature verification, 2 ms for normal processing)
Number of attackers	0-10

5.2 Simulation Results

Fig. 2-5 and Fig. 6-7 show the observed results under non-adversarial and adversarial scenarios, respectively. Each data point in the resulting graphs is an average of ten simulation runs with homogeneous configuration but different randomly generated mobility patterns.

As shown in Fig. 2-5, the performance of TSOR is almost identical to that of ARAN in the absence of attackers.

- From Fig. 2, the average packet delivery fraction decreases as the node speed increases. Both the protocols keep it above 97%.
- From Fig. 3, there is a steady increase in average routing load with the increase in node mobility. Compared to ARAN, TSOR has a very similar routing overhead.
- From Fig. 4, the average path length for both protocols is relatively stable between 3 to 4 hops.
- From Fig. 5, both ARAN and TSOR have the average route acquisition latency in the range of [90 ms, 125 ms]. Also, this metric keeps relative stable.

These simulation results above show that TSOR does not degrade the performance and achieves a very similar performance as that of ARAN in the scenario where there are no attackers.

Next, the performances of TSOR and ARAN are analyzed with respect to average path length and average route acquisition latency in the adversarial scenario when there are malicious nodes performing packet redirection and routing loop attacks. Fig. 6 and 7 compare TSOR with ARAN under packet redirection and routing loop attacks, respectively. Each data point in the graph reflects the scenario under different numbers of malicious nodes ranging from 2 to 10. The nodes speeds are averaged. As it can be observed, ARAN is greatly affected by these attacks.

- Fig. 6 shows that the average path length of ARAN is very high under the attacks. For instance, in the presence of routing loop attack when the number of attackers increases to 10, the average path length ARAN becomes two times as that of TSOR. However, these attacks have a very little impact on TSOR. The average path length in TSOR keeps stable in the range of 3 to 4 hops under each attack even there are 10 attackers. The results show that TSOR can resist packet redirection and routing loop attacks effectively. This is because TSOR can exclude the unreliable routes in the route reply phase before transmitting packets.
- Fig. 7 depicts that, the average route acquisition latency in TSOR is not affected by the attacks. It maintains a stable value that matches the non-adversarial scenario. However, ARAN suffers a lot. With the packet redirection attack and routing loop attack, when the number of attackers is up to 10, the route acquisition latency increase to 32% and 44%, respectively as compared to TSOR.

These results clearly show that TSOR provides better security in terms of finding reliable and shortest routes when compared with ARAN under the simulated attack models.

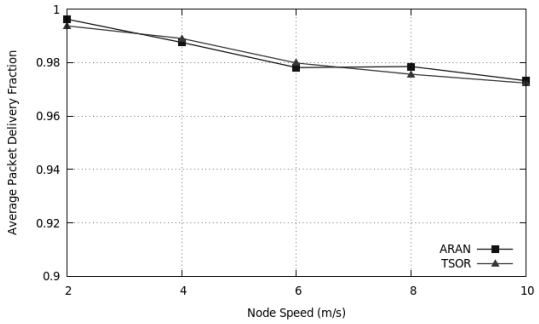

Fig. 2. Average packet delivery fraction

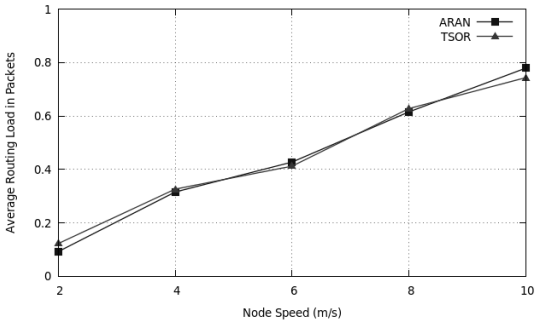

Fig. 3. Average routing load in packets

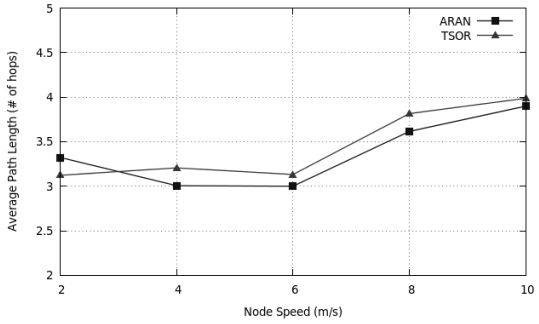

Fig. 4. Average path length

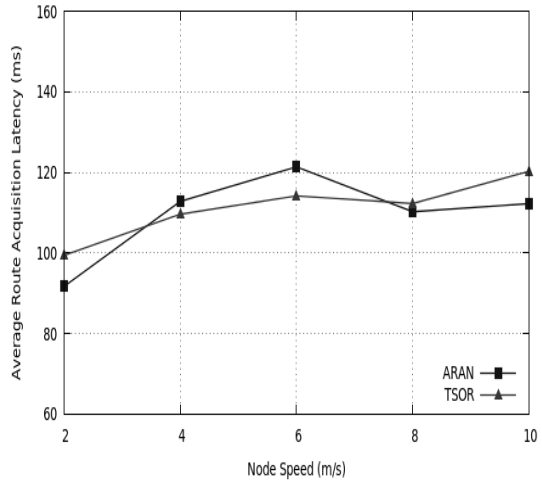

Fig. 5. Average route acquisition latency

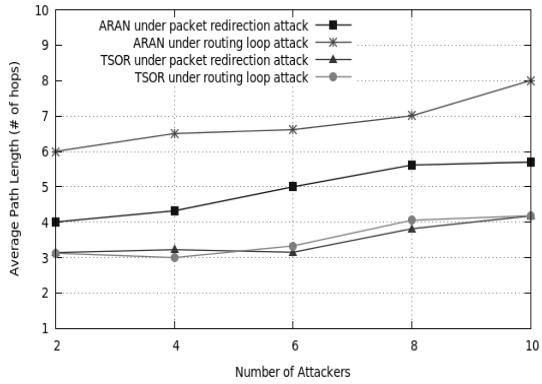

Fig. 6. Average path length under packet redirection and routing loop attack

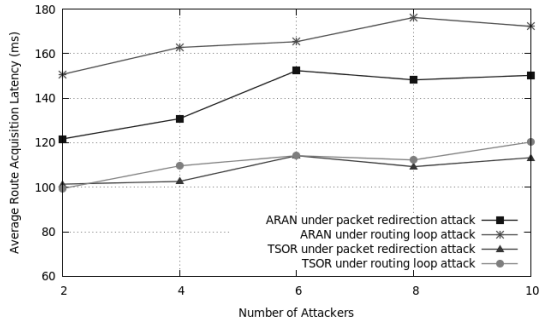

Fig. 7. Average route acquisition latency under packet redirection and routing loop attacks

6. CONCLUSION

In this paper, the design and analysis of a new secure on-demand routing protocol for MANET called Topology-hiding Secure On-demand Routing (TSOR) are presented. The protocol is based on the philosophy of topology hiding. The design of TSOR carefully fits the digital signature mechanism over a stringent hop-to-hop verification of packet traversal time in order to create an efficient and practical protocol that is robust against multiple active attacks. TSOR successfully archives all the security goals like packet integrity, authentication, availability, and non-repudiation. Performance evaluation shows that TSOR has a better capability of finding reliable and shortest under an adversarial scenario in comparison to the existing topology hiding on-demand secure routing protocols. However, TSOR does not degrade the performance when there is no attack. Together with existing approaches for securing the physical layer and MAC layer within the network protocol stack, TSOR provides a better approach for the secure operation of an ad hoc network.

7. REFERENCES

[1] R. Di Pietro, S. Guarino, N.V. Verde, J. Domingo-Ferrer, “Security in wireless ad-hoc networks – A survey”, Elsevier Computer Communications 51, 2014, pp.1–20.

[2] Soufiene Djahel, Farid Na¨ıt-abdesselam, and Zonghua Zhang, “Mitigating Packet Dropping Problem in Mobile Ad Hoc Networks: Proposals and Challenges”, IEEE Communications Surveys & Tutorials 13 (4), 2011, pp.658-672.

[3] W. Galuba, P. Papadimitratos, M. Poturalski, K. Aberer, Z. Despotovic, W. Kellerer, “Castor: scalable secure routing for ad hoc networks”, in: Proc. IEEE Conference on Computer Communications (INFOCOM), 2010, pp. 1-9.

[4] Loay Abusalah, Ashfaq Khokhar, and Mohsen Guizani, “A Survey of Secure Mobile Ad Hoc Routing Protocols”, IEEE Communications Surveys & Tutorials 10 (4), 2008 pp.78-93.

[5] Pervaiz, O. Mohammad, Mihaela Cardei, and Jie Wu, "Routing security in ad hoc wireless networks", Springer US Network Security, 2010, pp.117-142.

[6] N.M. Chacko, S. Sam, P.G.J. Leelipushpam, "A survey on various privacy and security features adopted in MANETs routing protocol", in: International Multi-Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s), 2013, pp.508-513.

[7] S. Kumar, G. Pruthi, A. Yadav, M. Singla, "Security Protocols in MANETs", in: Second International Conference on Advanced Computing & Communication Technologies (ACCT), 2012, pp.530-534.

[8] P. Papadimitratos, Z.J. Haas, "Secure data communication in mobile ad hoc networks", J. Select. Areas Commun. 24 (2), 2006, pp.343–356.

[9] Y.C. Hu, A. Perrig, D.B. Johnson, "Ariadne: a secure on-demand routing protocol for ad hoc networks", in: Proc. ACM International Conference on Mobile Computing and Networking (MOBICOM), Atlanta, 2002, pp.23-28.

[10] Jiang, Tingyao, Qinghua Li, and Youlin Ruan, "Secure dynamic source routing protocol", in: Proc. IEEE International Conference on Computer and Information Technology (CIT'04), 2004.

[11] M.G. Zapata, "Securing ad hoc routing protocols", in: Proc. ACM workshop on wireless Security, Atlanta, Sep 2002, pp.1-9.

[12] K. Sanzgir, B. Dahill, "A secure routing protocol for ad hoc networks", in: Proc. IEEE International Conference on Network Protocols, 2002, pp.1-10.

[13] Acs, Gergely, Levente Buttyan, and Istvan Vajda, "Provable security of on-demand distance vector routing in wireless ad hoc networks", Springer Security and Privacy in Ad-hoc and Sensor Networks, 2005, pp.113-127.

[14] S. Ghazizadeh, O. Ilghami, E. Sirin, "Security-aware adaptive dynamic source routing protocol", in: Proc. IEEE Conference on Local Computer Networks, 2002.

[15] Yujun Zhang, Tan Yan, Jie Tian, Qi Hu, Guiling Wang, Zhongcheng Li, "TOHIP: A topology-hiding multipath routing protocol in mobile ad hoc networks", Elsevier Ad Hoc Networks 21, 2014, pp.109–122.

[16] Yih-Chun Hu, A. Perrig, D.B. Johnson, "Packet leashes: a defense against wormhole attacks in wireless networks", in Proc. IEEE Conference on Computer Communications (INFOCOM), 2003, pp.1976-1986.

[17] http://www.isi.edu/nsam/ns.

[18] J. Broch, D.A. Maltz, D.B. Johnson, Y.C. Hu, J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols", in: Proc. ACM MOBICOM, Oct. 1998, pp.85–97.