A Secure Source Routing Protocol for Mobile Ad Hoc Networks

Baban A. Mahmood and D. Manivannan
University of Kirkuk and University of Kentucky

Routing protocols for Mobile Ad Hoc Networks (MANETs) have been extensively studied. Some of the well-known source routing protocols presented in the literature that claim to establish secure routes are susceptible to hidden channel attacks. In this paper, we address this issue and present a novel secure routing protocol, based on sanitizable signatures, that is not susceptible to hidden channel attacks.

Keywords: Routing in MANETs; Secure routing; Mobile ad hoc Networks; MANETs.

1. INTRODUCTION

Nodes in a Mobile Ad Hoc Network (MANET) form a network among themselves without the use of any fixed infrastructure such as access points or base stations, and communicate with each other by cooperatively forwarding packets on behalf of others. MANETs have applications in areas such as military, disaster rescue operations, monitoring animal habitats, etc. where establishing communication infrastructure is not feasible(Li and Singhal [2005], Shen and Zhao [2013], Talooki et al. [2013], Kavitha et al. [2013], Jain et al. [2013]).

Routing protocols designed for MANETs can be broadly classified as geographic routing protocols (e.g., GPSR Karp and Kung [2000]) and topology-based routing protocols. Topology-based routing can be classified into proactive routing protocols like DSDV proposed by Perkins and Bhagwat [1994] in which nodes use pre-established table-based routes, reactive (on-demand) routing protocols (e.g., AODV Perkins and Royer [1999]), and hybrid routing protocols(e.g., Li and Singhal [2005], Hsu and Lei [2009], Giruka and Singhal [2007]).

Due to their intrinsic characteristics, MANETs are more vulnerable to attacks than infrastructure based networks. There are two main categories of attacks, namely, passive attacks and active attacks (Abdelaziz et al. [2013]). In passive attacks, adversarial nodes do not disrupt the operation of routing protocols, whereas in active attacks, adversarial nodes try to disrupt route discovery, route maintenance and data forwarding. An adversarial node can disrupt the route discovery by offering a better route than a route offered by a genuine node to disrupt packet delivery. Adversarial nodes can also impersonate other nodes to launch other sophisticated attacks. One of the aims of a secure routing protocol is to establish a valid route between a source and a destination in the presence of adversarial nodes. Several secure routing protocols have been proposed in the literature for MANETs. Yih-Chun and Perrig [2004] present a survey of secure routing protocols.

Hurley-Smith et al. [2017] propose a security framework for routing in MANETs. They demonstrate the suitability of their framework for wireless communication security by comparing their framework with other frameworks such as IPsec through simulation. Xu et al. [2017] physical layer security aware routing protocols for ad hoc networks. Arulkumaran and Gnanamurthy [2019] propose a method for detecting black-hole attacks in MANETs using Fuzzy trust approach. Xia et al. [2019] propose an attack-resistant Trust inference model for securing routing in Vehicular Ad hoc NETworks (VANETs). Kojima et al. [2019] propose a secure routing protocol which improves security of an existing protocol. Li et al. [2019] present a method for finding routes

Authors' addresses: Baban A. Mahmood, Computer Science Department, University of Kirkuk, Baghdad St., Kirkuk, Iraq, *Email:Baban@netlab.uky.edu* and D. Manivannan, Department of Computer Science, University of Kentucky, Lexington, KY 40508, *Email: mani@cs.uky.edu*

to the destination through reliable nodes. They also demonstrate how it can resist black-hole attacks. Salehi and Boukerche [2019] point out how opportunistic routing protocols perform poorly in the presence of malicious nodes and classify the security-related opportunistic routing protocols into different categories and compare them.

1.1 Paper Objectives

In this paper, we observe that many of the routing protocols presented in the literature (e.g., SRP by Papadimitratos and Haas [2002], Ariadne by Hu et al. [2005], endairA by Ács et al. [2006], etc.) that are claimed to be secure were in fact proved to be susceptible to hidden channel attacks. Then, we propose a novel secure source routing protocol, called *Secure Ariadne (SAriadne)*, that establishes a secure and valid route. We discuss in detail how *SAriadne* is not susceptible to hidden channel attacks, unlike the protocols mentioned above.

1.2 Organization of the Paper

The rest of the paper is organized as follows: In Section 2, we review the related protocols and discuss how they have been identified to be susceptible to hidden channel attacks. In Section 3 we discuss *Chameleon Hash* functions and *Sanitizable Signatures* which are needed in the design of the proposed protocol. In Section 4, we present our protocol. In Section 5, we analyze and discuss how our protocol prevents hidden channel attacks. Section 6 concludes the paper.

2. RELATED WORK AND THEIR VULNERABILITIES

In this section, we discuss some of the secure source routing protocols presented in the literature and discuss how they are susceptible to hidden channel attacks.

2.1 Secure Routing in Mobile Ad Hoc Networks (SRP)

SRP, proposed by Papadimitratos and Haas [2002], is an on-demand source routing protocol that has the basic characteristics of reactive routing protocols. Route requests are generated by a source S and protected by Message Authentication Code (MAC)(Krawczyk et al. [1997]). The MAC is computed using the key that S shares with the target node T. S broadcasts the route request to all its neighbors. When a neighbor of S (or an intermediate node) receives the route request, it appends its id to the request and rebroadcasts the updated request if it has not seen that request before; otherwise, it discards the request. Intermediate nodes do not check the validity of the MAC in the request because no node possesses the key used to compute it except S and T.

When the target T receives the request, it verifies the MAC in the request. If the MAC is valid, then the target node assumes that all the adjacent pairs of nodes accumulated in the route request forms a valid route. The target then computes a MAC using the key it shares with the source and authenticates the route. This is then sent back to the source S through the reverse route traversed by the request. For example, a route request message received by an intermediate node X_j has the following form.

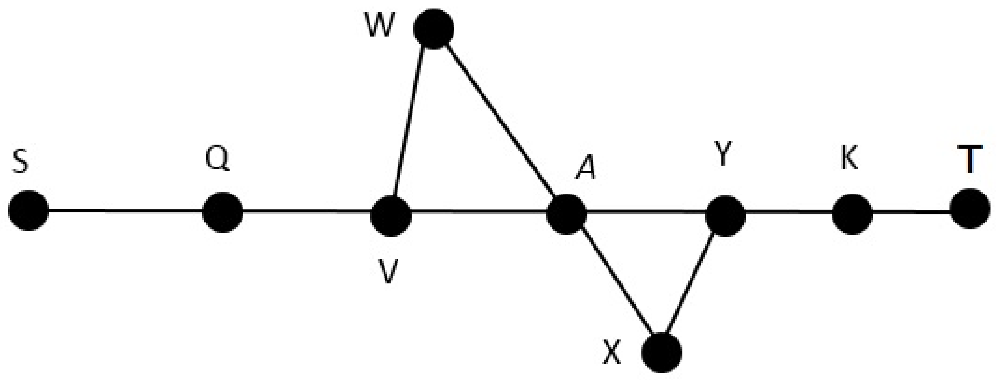
$$m_{rreq} = (rreq, S, T, id, sn, (X_1, ..., X_j), mac_S),$$

where id is the id assigned to the route request $rreq$, sn is a sequence number, and mac_S is the MAC computed on $rreq$, S, T, id, and sn by S using the key it shares with T. Now, if S, X_1, ..., X_p, T is the discovered route, then all intermediate nodes X_j, $(1 \leq j \leq p)$ will receive m_{rrep} as a route reply, where

$$m_{rrep} = (rrep, S, T, id, sn, (X_1, ..., X_p), mac_T),$$

where mac_T is the MAC computed by the target T using the key it shares with S on the message fields preceding it. Intermediate nodes must check the route reply header to verify that both *id* and *sn* fields are correct. In addition to that, intermediate nodes need to check that they are neighbors with both their upstream and downstream nodes before sending the route reply upstream.

An intrinsic point to observe in this protocol is that the upstream route from T to S is authenticated by the target, but the downstream route from S to T is not authenticated by S. This means that there could be malicious pair of nodes that are not neighbors but claim to be neighbors in the route request that reaches the target. These malicious nodes could divert traffic through other routes as observed by Papadimitratos and Haas [2002]. It is also possible for a malicious node to pad route requests with ids of nodes that are not its neighbors. The malicious node can impersonate these nodes in the reply phase in such a way that the reply propagates to the source node. Hence, the route received by the source may be invalid because some of the nodes specified in the route as neighbors may not be neighbors.



A Sample Network Configuration Wherein an Attack Exists Under SRP found by Buttyán et al.

2.1.1 *How SRP is Susceptible to Hidden Channel Attack.* Buttyán and Vajda [2004] have found that SRP is susceptible to hidden channel attack. We briefly discuss how SRP is susceptible to hidden channel attack using the network configuration in Figure 1; in this Figure, A is an adversarial node. Suppose node S in Figure 1 broadcasts a route request to discover a route to the target T. When V receives this request, it rebroadcasts the request. Thus, A receives the request m_V from V, where

$$m_V = (rreq, S, T, id, sn, (Q, V), mac_S);$$

here id is the request id, sn is the sequence number, and mac_S is the MAC computed by S using the key it shares with T. A, the adversarial node, then inserts an arbitrary sequence of node ids λ into the request and broadcasts the message m_A, in the name of X, where

$$m_A = (rreq, S, T, id, sn, (Q, V, W, \lambda, X), mac_S).$$

The adversarial node A cannot determine if the pair of nodes V and W and the pair of nodes X and Y are neighbors; however, since the nodes (i.e., V,W,X, and Y) in these pairs are neighbors of A, A makes a guess that V and W, as well as X and Y are neighbors. Y is a neighbor of both A and X and the node list ends with X; hence, when Y receives m_A, it appends its id to the node list in the request, and rebroadcasts the updated request m_Y, where

$$m_Y = (rreq, S, T, id, sn, (Q, V, W, \lambda, X, Y), mac_S).$$

When the target T receives the request from K, after it successfully verifies mac_S, T computes the MAC (i.e., mac_T) over the route and unicasts the reply m_T via the nodes in the route request in the reverse order, where

$$m_T = (rrep, S, T, id, sn, (Q, V, W, \lambda, X, Y, K), mac_T).$$

When the intermediate node Y forwards m_T to node X, node A overhears the transmission. A then forwards the message m_T to V in the name of W. Since W and V are neighbors, V believes the message is from W and forwards it to Q which in turn sends it to S. S then successfully verifies mac_T and accepts the route $(Q, V, W, \lambda, X, Y, K)$ which is clearly an invalid route. **Thus, SRP is susceptible to hidden channel attack.**

2.2 The Basic Ariadne Protocol

Hu et al. [2005] presented ***Ariadne***, a secure source routing protocol, based on Dynamic Source Routing protocol (DSR), proposed by Johnson and Maltz [1996]. Under this protocol, when a source S performs a route discovery for a target T, it is assumed that both S and T share the secret key K_{ST} and K_{TS} for authenticating messages. Ariadne imposes two requirements in the route discovery phase:

—First, that the target can authenticate each node in the route traversed by the route request so that it returns a route reply along a path that has legitimate nodes.

—Second, that a source node can authenticate each node in the route (i.e., the node list in the route reply).

To achieve node list authentication, the authors use three different techniques: the TESLA protocol, the standard MACs, and the digital signatures. TESLA, proposed by Perrig et al. [2000] is a broadcast authentication scheme that requires time synchronization. Unlike other asymmetric protocols such as RSA, proposed by Rivest et al. [1978], TESLA achieves the asymmetry from both clock synchronization and delayed key disclosure. However, TESLA depends on the ability of a receiver to correctly ***determine which keys a sender might have published (i.e., the TESLA security condition)***.

In the following subsections, we discuss the three techniques used in Ariadne to make it a secure routing protocol and discuss how all these three techniques have been found to be susceptible to hidden channel attacks.

2.3 Basic Idea behind Ariadne with Signature

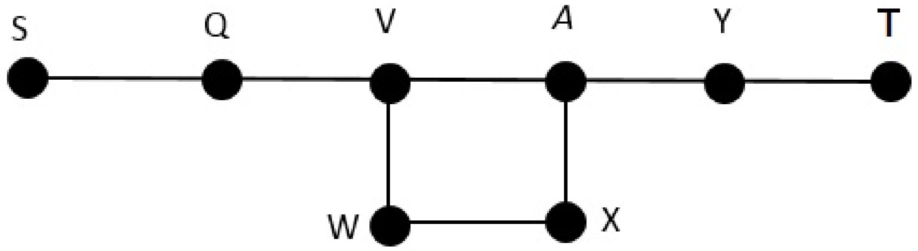
Ariadne with signatures proposed by Hu et al. [2005] differs from SRP in two aspects, as noted by Buttyán and Vajda [2004]. First, in addition to source and target nodes, intermediate nodes include their own digital signatures in route requests. Second, per-hop hashing is used to prevent removal of legitimate nodes from the node list in a route request. A source node broadcasts a route request message to its neighbors. The route request contains the source id S and the target id T, a request id, and the MAC computed over these elements using the key S shares with T. Together with its own id, each intermediate node hashes the MAC using a one-way hash function. These hash values computed by intermediate nodes are called per-hop hash values. An intermediate node then appends its id to the node list accumulated in the route request and generates a digital signature over this updated request. This signature is then appended to the signature list in the request and the request is re-broadcast. For example, as noted by Buttyán and Vajda [2004], a route request message forwarded by an intermediate node X_j has the following form

$$m_{rreq} = (rreq, S, T, id, h_{X_j}, (X_1, ..., X_j), (sig_{X_1}, ..., sig_{X_j})),$$

where S and T are the source and target identifiers respectively, id is the request identifier, h_{X_j}

is the per-hop hash value calculated by X_j, $(X_1, ..., X_j)$ is the node list, and $(sig_{X_1}, ..., sig_{X_j})$ is the signature list. When the route request arrives at the target T, T verifies the MAC attached by the source, the per-hop hash of each intermediate node, and individual signature of each intermediate node in the signature list. If these verifications are successful, the target generates a route reply and sends it back through the list of nodes in the request in the reverse order. Each intermediate node forwards the reply to the next hop without modifying the reply. The reply contains the source id and the target id, the accumulated route and the digital signatures of the intermediate nodes obtained from the route request, and a digital signature that the target computed over these elements.

When the source S receives the reply, it verifies the digital signature of the target and the individual signature of each intermediate node. S accepts the route returned in the route reply if all these verifications are successful.



A Sample Network Configuration Wherein an Attack Exists on Ariadne with Signatures as Described by Buttyán and Vajda.

2.3.1 *Attack on Ariadne with Signatures.* Buttyán and Vajda [2004] discovered how Ariadne with signatures, presented above, is susceptible to hidden channel attack. We use Figure 2, which shows part of the network configuration, to illustrate this attack; in this network, the adversarial node A launches an attack on the route discovery initiated by S to discover a route to T. When A receives the request m_V from node V, where

$$m_V = (rreq, S, T, id, h_V, (Q, V), (sig_Q, sig_V)),$$

it does not rebroadcast the request. Also, A receives request m_X from node X, where

$$m_X = (rreq, S, T, id, h_X, (Q, V, W, X), (sig_Q, sig_V, sig_W, sig_X)).$$

A obtains h_V from m_V and the signatures sig_Q, sig_V, and sig_W from m_X. From m_X, A knows that W and V are neighbors. A then computes its per-hop hash value (i.e., $h_A = H(A, H(W, h_V))$), where H is the publicly known hash function. A then uses this information to generate and broadcast the request m_A, where

$$m_A = (rreq, S, T, id, h_A, (Q, V, W, A), (sig_Q, sig_V, sig_W, sig_A)).$$

When the target T receives the request, T verifies the signatures and creates a route reply m_T and sends it back to S, where

$$m_T = (rrep, T, S, (Q, V, W, A, Y), (sig_Q, sig_V, sig_W, sig_A, sig_Y, sig_T)).$$

When A receives m_T, it forwards it to V in the name of W which in turn forwards it to Q.

When S receives the reply, it validates all the signatures and accepts the route (Q, V, W, A, Y), which is an invalid route because A and W are not neighbors. **Thus, Ariadne with signature protocol is susceptible to hidden channel attack.**

2.4 Basic Idea Behind Ariadne with MAC

Ariadne with MAC presented by Hu et al. [2005] is similar to Ariadne with signature, presented in Section 2.3 with the exception that it does not use digital signatures. In the route discovery phase, the source node S broadcasts a route request message to find a route to the target T. The request message contains the ids of both the source and the target, a request id, and the MAC that is computed over these elements. The MAC is computed by S using the key (K_{ST}) it shares with T. When an intermediate node X receives the route request, X hashes the MAC with its own id using a one-way hash function. These hash values are called per-hop hash values to prevent an intermediate node from removing nodes from the node list in the route request, as the route request propagates.

When an intermediate node receives the request for the first time, it computes the per-hop hash value, appends its id to the accumulated node list contained in the request, and computes its own MAC on the updated request using the key it shares with T; it then appends its MAC to the MAC list in the route request and re-broadcasts the updated request. For example, as observed and presented by Ács et al. [2006], a route request message m_{rreq} received by an intermediate node X_j has the following form

$$m_{rreq} = (rreq, S, T, id, h_{X_j}, (X_1, ..., X_j), (mac_{X_1}, ..., mac_{X_j})),$$

where id is the request id, h_{X_j} is the per-hop hash value (i.e., hash chain) calculated by X_j, $(X_1, ..., X_j)$ is the node list, and $(mac_{X_1}, ..., mac_{X_j})$ is the MAC list; in this,

$$h_{X_j} = H(X_j, H(X_{j-1}, H(..., H(X_1, h_S)))),$$

and h_S is the hash value computed by S, where

$$h_S = MAC_{K_{ST}}(S, T, id).$$

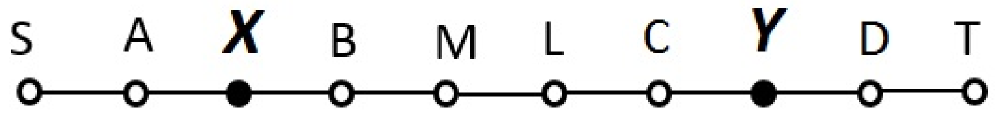
When the target T receives the route request, it verifies the hash values and the MAC attached by the source and the intermediate nodes. If these verifications are successful, the target generates a route reply and unicasts it back via the reverse route obtained from the route request. The route reply contains the id of the source and target, the node list obtained from the request, and a MAC computed by the target over these elements. The MAC attached by the target is computed using the key shared by the target and the source. Intermediate nodes do not modify the reply. Route reply m_{rrep} created by the target T is of the following form

$$m_{rrep} = (rrep, S, T, (X_1, ..., X_j), mac_T).$$

When the source S receives the route reply m_{rrep}, it only verifies the target's MAC mac_T. S accepts the route returned in the reply if this verification is successful. Otherwise, it discards the reply.

2.4.1 *Attack on Ariadne with MAC.* Ács et al. [2006] found that Ariadne with MAC, presented above, is susceptible to hidden channel attack. Consider the network configuration shown in Figure 3, wherein S initiates a route request to find a route to the target T, and X and Y are adversarial nodes that collude. X, the first adversarial node, receives the request m_A, where

$$m_A = (rreq, S, T, id, h_A, (A), (mac_A)).$$



A Sample Network Configuration wherein an Attack Exists on Ariadne with MAC, as presented by Ács et al.

Instead of appending its MAC, X puts h_A on the MAC list and rebroadcasts the request m_X. X does that because Y will need this hash value to omit the nodes between X and Y.

$$m_X = (rreq, S, T, id, h_A, (A, X), (mac_A, h_A)).$$

Because intermediate nodes do not verify the MACs, the intermediate node B does not detect this attack. The second adversarial node Y receives the route request m_C, where

$$m_C = (rreq, S, T, id, H(C, H(L, H(M, H(B, h_A)))),\\ (A, X, B, M, L, C), (mac_A, h_A, mac_B, mac_M, mac_L, mac_C)).$$

Then, Y drops the nodes (B, M, L, C) from the node list and their corresponding MACs from the MAC list (i.e., $(mac_B, mac_M, mac_L, mac_C)$). Y can do that because it knows that the request has passed through the first adversarial node X by recognizing X in the node list. Y also could get h_A from the MAC list by determining the position of X in the node list. Hence, Y computes $h_Y = H(Y, h_A)$, needed to omit the nodes (B, M, L, C), and its MAC mac_Y on the modified request. Then Y rebroadcasts the modified request m_Y that is received, updated, and rebroadcast by D, where

$$m_Y = (rreq, S, T, id, h_Y, (A, Y), (mac_A, mac_Y)).$$

When T receives the request, it validates the MACs and the per-hop hash value in the request, generates a reply m_T, and sends it back to S, where

$$m_T = (rrep, T, S, (A, Y, D, T), mac_T).$$

When Y receives m_T, it re-inserts the dropped nodes into the node list and forwards the modified reply m_{Yrrep} to C, where

$$m_{Yrrep} = (rrep, T, S, (A, X, B, M, L, C, Y, D, T), mac_T).$$

Since the intermediate nodes B, M, L, C do not verify m_{Yrrep}, each of them forwards the reply m_{Yrrep}. When X receives the reply, it removes the nodes B, M, L, C and its id from the list and forwards the reply $m_{X_{rrep}}$ to A, where

$$m_{X_{rrep}} = (rrep, T, S, (A, Y, D, T), mac_T).$$

When S receives $m_{X_{rrep}}$, it computes the MAC over the fields preceding the mac_T in route reply and verifies if it is same as mac_T that was computed by T; if it is, then it accepts the route as a valid route, even though it is an invalid route. **Thus, the Ariadne with MAC protocol is susceptible to hidden channel attack.**

2.5 Basic Idea Behind the Optimized Version of Ariadne with Iterated MAC

Ariadne has another version that uses iterated MAC computations (Hu et al. [2003]), instead of independent MACs that are computed separately, as in section 2.4. As noted by Ács et al. [2006], compared to the other versions of Ariadne, this iterated MAC version has superior security characteristics and is more secure. Like the other versions, a source node broadcasts a route request to all its neighbors. Each intermediate node updates the request that is received for the first time and re-broadcasts the updated request. The route request m_j that reaches an intermediate node X_j, $(1 \leq j \leq p$, on a route $S = X_0$,X_1,...,X_p,$X_{p+1} = T$ is of the following form

$$m_j = (rreq, S, T, id, (X_1, ..., X_j), mac_{SX_1 \cdots X_j}),$$

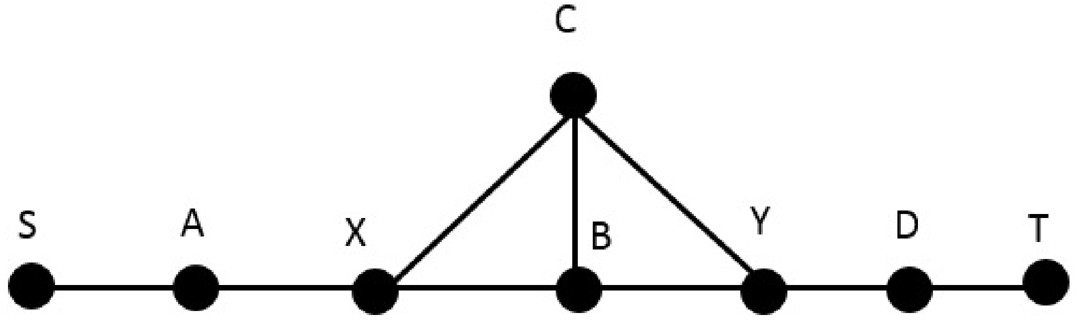
where id is the request id, $(X_1, ..., X_j)$ is the accumulated route (i.e., the node list), $mac_{SX_1 \cdots X_j}$ is the MAC computed by X_j with the key it shares with the target T over the route request m_{j-1} received from X_{j-1}, where

$$m_{j-1} = (rreq, S, T, id, (X_1, ..., X_{j-1}), mac_{SX_1 \cdots X_{j-1}}).$$

When the target T receives the route request from the last intermediate node X_p, it recomputes the MAC iteratively and compares it with the value received in the request, if they are same. Since T shares a key with each intermediate node, it can iteratively reconstruct the MAC sequence. If the verification is successful, it means that each intermediate node in the node is genuine. The target then generates a route reply m_{rrep}, where

$$m_{rrep} = (rrep, S, T, id, (X_1, ..., X_p), mac_T);$$

here mac_T is the MAC computed by T with the key it shares with S on the message fields that precede it (i.e., on $(rrep, S, T, id, X_1, ..., X_p)$). The reply m_{rrep} is then unicast via the nodes in the request in the reverse order (i.e., via the nodes $X_p, X_{p-1}, ..., X_1$) to the source S. When an intermediate node receives the reply, it verifies that its id is in the node list. It also verifies that the id preceding it and the id next to it in the node list are its neighbors. Intermediate nodes do not modify the reply. When the source receives the route reply m_{rrep}, it accepts the route returned in m_{rrep} as a valid route if it can successfully verify the MAC computed by the target, i.e., mac_T.



A Sample Network Configuration wherein an Attack on the Optimized Version of Ariadne with Iterated MAC is Possible.

2.5.1 *Susceptibility of Optimized Version of Ariadne with Iterated MAC to Hidden Channel Attack.* Susceptibility of Optimized Version of Ariadne with Iterated MAC to hidden channel attack was first discovered and presented by Ács et al. [2006], and briefly described by Burmester and de Medeiros [2009]. Consider the network configuration in Figure 4, in which the source S needs to find a route to the target T, and X and Y are adversarial nodes that collude and launch an attack on the route discovery. The first adversarial node X receives the route request m_A from the node A, where

$$m_A = (rreq, S, T, id, (A), mac_{SA}.)$$

X computes the MAC (i.e., mac_{SAX}) over m_A after appending its id to the node list in the request and then broadcasts the updated request m_X, where

$$m_X = (rreq, S, T, id, (A, X), mac_{SAX}).$$

When B and C receive m_X, they update the request and broadcast the corresponding route request. Y does not respond to the requests coming from B and C. A little later, X creates a route reply $m_{X_{rrep}}$ in the name of Y and unicasts it to B, where

$$m_{X_{rrep}} = (rrep, S, T, id, (A, X, B, Y), mac_{SAX}).$$

Note that the MAC in this fake reply is wrong (i.e., it was not computed by the target). B only checks the id of the reply and whether X and Y are its neighbors. Since B has previously received request m_X with id same as that included in this reply $m_{X_{rrep}}$, it retransmits $m_{X_{rrep}}$ to X. Y intercepts $m_{X_{rrep}}$ whose MAC is mac_{SAX} which is needed by Y to remove B. Y then creates and broadcasts route request m_Y, where

$$m_Y = (rreq, S, T, id, (A, X, Y), mac_{SAXY}).$$

Node D receives this request, appends its id to the node list, appends its MAC, and rebroadcasts the updated request m_D, where

$$m_D = (rreq, S, T, id, (A, X, Y, D), mac_{SAXYD}).$$

T receives m_D and verifies the iterated MAC mac_{SAXYD}. Since mac_{SAXYD} was correctly constructed, T accepts it, generates a route reply m_T, and sends it back to the source S, where

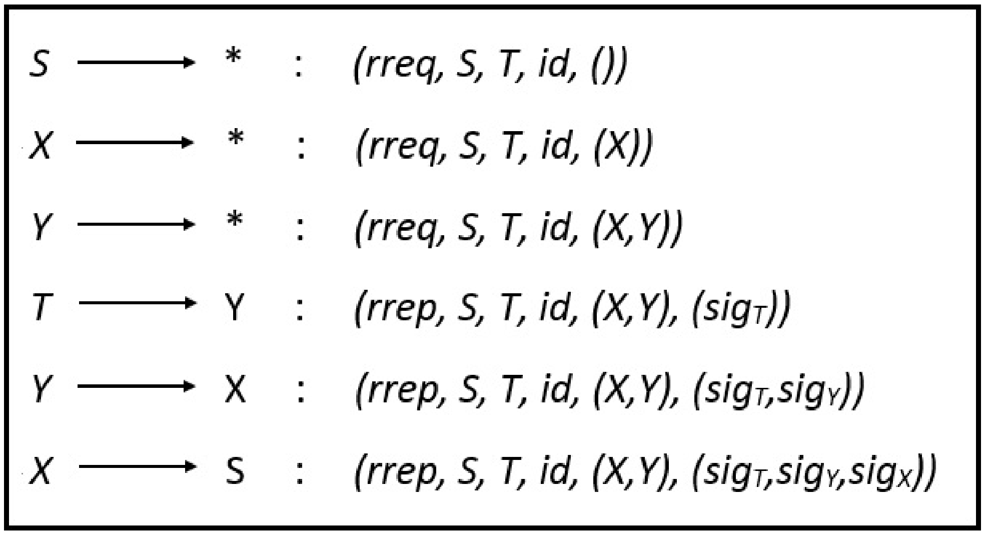
$$m_T = (rrep, S, T, id, (A, X, Y, D), mac_T).$$

When D receives m_T, it accepts it because D's id is in the list and both T and Y are neighbors of D. When Y receives m_T from D, it adds the id for C to the node list and sends the modified message $m_{Y_{rrep}}$ to C, where

$$m_{Y_{rrep}} = (rrep, S, T, id, (A, X, C, Y, D), mac_T).$$

Node C accepts the message and forwards it to X. When X receives $m_{Y_{rrep}}$, it removes the id of C from the node list and forwards the modified message to S which in turn successfully verifies the reply to contain a valid route. However, the reply does not contain a valid route.

In this attack, to remove the node B or C from the node list, the second adversarial node Y needed the mac_{SAX} which was computed by X in the message $m_{X_{rrep}}$. Y needs this MAC in order to compute mac_{SAXY}. Hence, the adversarial nodes X and Y successfully shortened the route to (A, X, Y, D) which is invalid because X and Y are not neighbors. **Thus, the optimized version of Ariadne with iterated MAC is susceptible to hidden channel attack.**



$S \longrightarrow * \;:\; (rreq, S, T, id, ())$

$X \longrightarrow * \;:\; (rreq, S, T, id, (X))$

$Y \longrightarrow * \;:\; (rreq, S, T, id, (X,Y))$

$T \longrightarrow Y \;:\; (rrep, S, T, id, (X,Y), (sig_T))$

$Y \longrightarrow X \;:\; (rrep, S, T, id, (X,Y), (sig_T, sig_Y))$

$X \longrightarrow S \;:\; (rrep, S, T, id, (X,Y), (sig_T, sig_Y, sig_X))$

An Illustration of the Operation of endairA. S is the source, T is the target, and X and Y are the intermediate nodes. id is a request id. sig_X, sig_Y, and sig_T are digital signatures of X, Y, and T, respectively. Each signature is computed over the message fields preceding it (including the previous signatures).

2.6 Basic Idea Behind endairA Protocol

endairA, presented by Ács et al. [2006], is a secure source routing protocol, in which the route reply is authenticated; hence intermediate nodes sign the route reply instead of the route request. A source S broadcasts a route request that contains the identifiers of both the source and the target and a request id. Intermediate nodes that receive the request for the first time append their id to the node list and rebroadcast the request. A typical route request m_j that is broadcast by an intermediate node X_j, $(0 \leq j \leq i)$, on a route $S = X_0, X_1, ..., X_i, X_{i+1} = T$, is of the form

$$m_j = (rreq, S, T, id, (X_1, ..., X_j)).$$

When the target receives the route request, it creates a route reply that contains the ids of the source and the target, the node list found in the request, and a digital signature computed by the target over the elements preceding the signature. The reply is unicast upstream through the nodes in the request in the reverse order. When an intermediate node receives the reply, it performs the following verifications:

(1) its id is in the node list in the reply.

(2) the id preceding it and the id next to it in the node list are its neighbors.

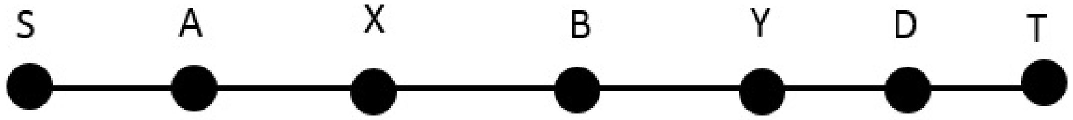
(3) the signatures in the route reply are valid.

If these verifications succeed, the intermediate node signs the reply and forwards it to the next node in the node list towards the source. As Ács et al. [2006] present, a route reply, unicast by X_j, $(0 \leq j \leq i)$, is of the form

$$m_j = (rrep, S, T, id, (X_1, ..., X_i), (sig_T, ..., sig_{X_j})),$$

where sig_{X_j} is the digital signature of X_j on the message fields preceding it.

When the source S receives the route reply, it checks if X_1 (i.e., the first id in the node list) is one of its neighbors. Then it verifies each individual signature in the reply. S accepts the route returned in the reply if all these verifications succeed. Figure 5 illustrates the operation of endairA.



A Sample Network Configuration Contains Adversarial Nodes X and Y.

2.6.1 *Attack on endairA.* Burmester and de Medeiros [2009] describe how endairA, presented by Ács et al. [2006], which we discussed above, is also susceptible to hidden channel attack. To illustrate this attack, consider the network configuration shown in Figure 6 in which the source S needs to find a route to the target T. *In this configuration, nodes X and Y are adversarial nodes and they collude to remove B from the node list in the route request as follows.* Under endairA, the second adversarial node Y receives the route request m_B from node B, where

$$m_B = (rreq, S, T, id, (A, X, B)).$$

Since endairA does not authenticate route requests, Y drops node B from the node list contained in m_B and broadcasts the modified route request m_Y,where

$$m_Y = (rreq, S, T, id, (A, X, Y)).$$

When T receives the route request, it generates a route reply and unicasts it back to S. The second adversarial node Y receives the reply $m_{D_{rrep}}$ from D, where

$$m_{D_{rrep}} = (rrep, S, T, id, (A, X, Y, D), (sig_T, sig_D)).$$

If Y signs the reply and forwards it to B, B will discard it because its id is not in the node list. Also, S validates the route reply only if every node in the node list returned in route reply has signed the reply. Therefore, the first adversarial node X needs the signature list (sig_T, sig_D, sig_Y). Since X and Y are adversarial nodes, they could have shared their private information. This means that X can reconstruct the digital signature of Y (i.e., sig_Y). Now, Y needs to find a mechanism through which it can deliver the signatures sig_T and sig_D to X. Suppose that the node D had previously issued a route request $m_{D_{rreq}}$ with a request id $id^{'}$ to discover a route to node A. Therefore, Y must have received $m_{D_{rreq}}$ from D

$$m_{D_{rreq}} = (rreq, D, A, id^{'}, ()).$$

Since route requests are not secured, Y can exploit the previous request from D to A to deliver the signatures (i.e., (sig_T and sig_D)) to X. To do that, Y modifies the id $id^{'}$ of the request $m_{D_{rreq}}$ into some other id $id^{''}$ such that it contains information that X can use to reconstruct the signatures sig_T and sig_D. *This information can also be encrypted.* Intermediate nodes do not detect this modification and rebroadcast the request. When X receives this request, it reconstructs the signatures, generates route reply m_X, and forwards it to A which in turn signs the reply and forwards it to S, where

$$m_X = (rrep, S, T, id, (A, X, Y, D), (sig_T, sig_D, sig_Y, sig_X)).$$

The source S then verifies that each individual signature was calculated correctly by the corresponding nodes and accepts (A,X,Y,D) as a valid route. **Thus, endairA protocol is also susceptible to hidden channel attack.**

3. PRELIMINARIES

In this section, we review Chameleon Hash functions and Sanitizable Signatures that are needed in the design of our protocol.

3.1 Chameleon Hash Functions without Key Exposure

Chameleon hash functions have the same characteristics of any cryptographic hash function with a trapdoor property, which allows the computation of collisions and second pre-images once the trapdoor information is known.

Ateniese and Medeiros [2004] proposed chameleon hash with key exposure freeness property. This key exposure free chameleon hash function is defined by the following set of algorithms:

—*Key Generation (KeyGen)*: A probabilistic algorithm that accepts a security parameter κ as input, and outputs a (secret key, public key) pair (sk,pk) as follows

$$\textbf{KeyGen} : 1^{\kappa} \longrightarrow (sk, pk))$$

—*Hash*: A function that accepts a public key pk, a label $\mathcal{L}$, a message m and an auxiliary random parameter r as input, and outputs a bitstring C of fixed length τ as follows

$$\textbf{Hash} : (pk, \mathcal{L}, m, r) \longrightarrow C \in \{0, 1\}^{\tau}$$

—*Universal Forge (Uforge)*: A function that accepts the secret key sk (associated with public key pk), a label $\mathcal{L}$, a message m and an auxiliary parameter r as input, and computes another message m' and random parameter r' as follows

$$\textbf{UForge}(sk, \mathcal{L}, m, r) \longrightarrow (m', r')$$

such that

$$\textbf{Hash}(pk, \mathcal{L}, m, r) = C = \textbf{Hash}(pk, \mathcal{L}, m', r')$$

—*Instance Forge (IForge)*: A function that accepts a public key pk, a label $\mathcal{L}$, and two pairs of a message and an auxiliary random parameter (m, r, m', r') as input, and computes another collision message m'' and random parameter r'' as follows

$$\textbf{IForge}(pk, \mathcal{L}, m, r, m', r') \longrightarrow (m'', r'')$$

such that

$$\textbf{Hash}(pk, \mathcal{L}, m, r) = C = \textbf{Hash}(pk, \mathcal{L}, m', r') = \\ \textbf{Hash}(pk, \mathcal{L}, m'', r'')$$

The above mentioned functions ensure the following security requirements:

—*Collision-Resistance,* which means that given only pk, $\mathcal{L}$, m and r, there is no efficient algorithm that can find a second pair m', r' such that $C = Hash(pk, \mathcal{L}, m, r) = Hash(pk, \mathcal{L}, m', r')$.

—*Semantic Security,* which means that the chameleon hash value C does not reveal anything about the message m that was hashed.

—*Key Exposure Freeness,* which means that given $C = Hash(pk, \mathcal{L}, m, r)$ there is no efficient algorithm that can find a collision (i.e., a second pair m', r' mapping to the same digest C) if a recipient with public key pk has never computed a collision under label $\mathcal{L}$.

Now, we describe the algorithms which create practical instances by using cryptographic primitives from a discrete log-based (DL) cryptosystem to construct an efficient chameleon hash trapdoor commitment scheme providing key exposure freeness, as presented by Ateniese and Medeiros [2004].

The scheme consists of three efficient algorithms: Key Generation Algorithm, Hash Calculation Algorithm, and Collision Finding Algorithm.

3.1.1 *Key Generation Algorithm.* An algorithm that takes a security parameter λ as input and outputs system public parameters $\mathsf{params} = \langle p, q, g, H \rangle$, where

—p and q are primes such that $p = 2q + 1$,

—g is a generator of the subgroup of quadratic residues Q_p of $\mathbb{Z}_p^*$,

—and $H : \{0, 1\}^* \mapsto \{0, 1\}^\tau$ is a collision-resistant hash function mapping arbitrary-length bit-strings to strings of fixed length τ.

The recipient chooses as secret key x at random in [1,q-1], and then computes the corresponding public key as $y = g^x$.

3.1.2 *Hash Calculation Algorithm.* An algorithm that takes a public key y, message m, and random values $(r,\ s) \in \mathbb{Z}_q$ x $\mathbb{Z}_q$ and then computes $e = H(m, r)$ and $C \leftarrow Hash(m, r, s) = r - (y^e g^s\ mod\ p)\ mod\ q$.

3.1.3 *Collision Finding Algorithm.* An algorithm that takes a hash output C, a random message m', and a random value $k' \in [1, q-1]$ and then computes a collision $(m',\ r',\ s')$ as follows
$r' = C + (g^{k'}\ mod\ p)\ mod\ q$
$e' = H(m', r')$
$s' = k' - e'^x\ mod\ q$
such that $C = Hash(m, r, s) = Hash(m', r', s')$

3.2 Sanitizable Signatures

The Sanitizable Signature scheme proposed by Ateniese et al. [2005] allows a node to **modify** designated portions of the document and **produce** a valid signature on the legitimately modified document **without any help from the original signer**. The designated portions are indicated as mutable which are subject to a prior agreement between the signer and the node. A sanitizable signature scheme is said to be weakly transparent if verifiers can identify which parts of the message are potentially sanitizable and, consequently, which parts are immutable.

A sanitizable signature scheme, presented by Ateniese et al. [2005] is defined by the following set of algorithms:

—*Key Generation Algorithm*: A probabilistic algorithm to compute the Signer's two public-private key pairs: pk^S_{sign} and sk^S_{sign} (for a standard digital signature algorithm) and the Target's two public-private key pairs pk^T_{sanit} and sk^T_{sanit} (for sanitization steps).

—*Signing Algorithm*: A deterministic algorithm which takes as input of a message m, a private signing key sk^S_{sign}, a public sanitization key pk^T_{sanit}, random coin r, and produces a signature σ as follows

$$\sigma \leftarrow SIGN(m, r; sk^S_{sign}, pk^T_{sanit})$$

—*Verification Algorithm*: A deterministic algorithm that, on input a message m, a possibly valid signature σ on m, a public signing key pk^S_{sign} and a sanitization key pk^T_{sanit}, outputs TRUE or FALSE as follows

$$VERIFY(m, \sigma; pk^S_{sign}, pk^T_{sanit}) \rightarrow \{TRUE, FALSE\}$$

—*Sanitization Algorithm*: A deterministic algorithm that, on input of a message m, a signature σ on m using public signing key pk^S_{sign}, a private sanitizing key sk^T_{sanit}, and a new message m', produces a new signature σ' on m' as follows

$$\sigma' \leftarrow SANIT(m, \sigma, m'; pk^S_{sign}, sk^T_{sanit})$$

The above mentioned scheme meets the following security requirements:

—*Correctness* which means that the verification algorithm must accept a signature produced by the signing algorithm.

—*Unforgeability* which means that it is difficult to produce a valid signature on a message that verifies against the associated public key without the knowledge of the private signing key.

—*Identical Distribution* which means that σ values produced by the sanitization algorithm are distributed identically to those produced by the signing algorithm.

4. THE PROPOSED PROTOCOL

In this section, we present our secure on-demand routing protocol, called *SAriande*, based on Sanitizable Signatures discussed above. This protocol prevents the *Hidden Channel Attacks.*

4.1 Protocol Setup

In our protocol, we use the sanitizable signature proposed by Ateniese et al. [2005], constructed based on the chameleon hashes presented by Ateniese and Medeiros [2004]. The two pairs of keys used in our protocol are the (private, public) signing keys (sk^S_{sign}, pk^S_{sign}) of the source S, and the (private, public) sanitizing keys (sk^T_{sanit}, pk^T_{sanit}) of the target T. To avoid confusion, $SIGN(.)$ is used for the sanitizable signature and $Sig(.)$ is used for the underlying signature algorithm. We assume that every node has the public key of both the signer and sanitizer. The basic idea behind the route request propagation and the route reply propagation of our protocol is similar to that of the optimized version of Ariadne, but we prevent hidden channel attacks using sanitizable signature.

4.2 Basic Idea Behind our Protocol

We use *sanitizable signature*, proposed by Ateniese et al. [2005], in our protocol to detect and prevent hidden channel attacks. The source node signs the route request using its private key sk^S_{sign} and the public sanitizing key pk^T_{sanit} of the target. This sanitizable signature is weakly transparent as described by Ateniese et al. [2005] to all the nodes in the network. It will be sanitized by the target T using T's private sanitizing key sk^T_{sanit}.

Since this signature is weakly transparent, every other node can verify the modifications to the sanitizable portion of the message. This helps every node verify whether the reply is coming from the target node T. Hence, no malicious node can impersonate the target T. This prevents the hidden channel attack during the propagation of the route request to the target node.

4.2.1 *Basic Route Discovery.* When a source node S needs to discover a route to a target T, it broadcasts a route request. To do that, S generates a message m and a signature σ on m using an algorithm $SIGN$, described in section 3.2. The route request message M_S, initiated by

S and targeted to T, contains eight fields: *(route request, message m, sanitizable signature σ, source, target, request id, node list, MAC list).* The *source* and the *target* fields are set to the ids of the source and the target nodes, respectively. The *request id* identifies the route request. As in Ariadne, proposed by Hu et al. [2005], the source S initializes the *node list* and *MAC list* to empty lists. S creates a message $m = (m_1, ..., m_t)$ that is partitioned into t blocks for some constant t and initializes m to *(request id, message id, source, target, sanitizable-portion of the message)*, where

$$m = (id, m_{ID}, S, T, sanitizable - portion).$$

The source S decides which portions, say $m_{i_1}, ..., m_{i_k}$, can be modified by the target T. The source computes the chameleon hash, $CH_{pk^T_{sanit}}(.)$, using the target's public key pk^T_{sanit} for all m_i as follows:

$$\bar{m}_i = \begin{cases} CH_{pk^T_{sanit}}(m_{ID}||i||m_i, r_i) \text{if } i \in \{i_1, i_2, ..., i_k\}, \\ m_i||i \text{ otherwise} \end{cases}$$

To sign $\bar{m}$, where $\bar{m} = (\bar{m}_1, \bar{m}_2, ..., \bar{m}_t)$, S uses a *Signing Algorithm* similar to that presented in Section 3.2. S signs $\bar{m}$ with its private signing key sk^S_{sign} and the public sanitization key pk^T_{sanit} of the target T. The $SIGN$ algorithm takes $\bar{m}$, the private signing key of the source sk^S_{sign}, the public sanitization key of the target pk^T_{sanit}, and a random coin r as input and produces the signature σ, where

$$\sigma = SIGN(\bar{m}, r; sk^S_{sign}, pk^T_{sanit}) = Sig_{sk^S_{sign}}(m_{ID}||t||pk^T_{sanit}||\bar{m}_1||...||\bar{m}_t).$$

Then S broadcasts the route request M_S, where

$$M_S = (rreq, \bar{m}, \sigma, S, T, id, (), ()).$$

The route request M_j forwarded by an intermediate node X_j $(1 \leq j \leq n)$, on the route $S = X_0$,X_1,...,X_n,$X_{n+1} = T$ is of the form

$$M_j = (rreq, \bar{m}, \sigma, S, T, id, (X_1, ..., X_j), (mac_{SX_1 \cdots X_j})).$$

$mac_{SX_1 \cdots X_j}$ is the MAC computed by X_j, with the key it shares with T, on the route request M_{j-1} received from X_{j-1}, where

$$M_{j-1} = (rreq, \bar{m}, \sigma, S, T, id, (X_1, ..., X_{j-1}), (mac_{SX_1 \cdots X_{j-1}})).$$

When the intermediate node X_j receives the route request that is intended to a target T, X_j verifies the signature σ using a Verification Algorithm, *VERIFY*, explained in section 3.2, where

$$VERIFY(\bar{m}, \sigma; pk^S_{sign}, pk^T_{sanit}) \longrightarrow \{TRUE, FALSE\}.$$

This verification is necessary to prevent the hidden channel attack on endairA mentioned in Section 2.6.1. If the signature is valid, X_j checks whether it has received the same request by checking the *source id*, *target id*, and *request id*. It discards the request if it has been received before. If not, X_j appends its id X_j to the *Node List* and replaces the $mac_{SX_1 \cdots X_{j-1}}$ with $mac_{SX_1 \cdots X_j}$ as explained above; X_j then rebroadcasts the route request M_j.

When the target T receives the route request, first it verifies the signature σ using the verification algorithm *VERIFY*, and then it validates the nodes in the list by recomputing the $MACs$.

If these verifications are successful, T sanitizes the message $\bar{m}$ using its private sanitization key sk^T_{sanit} (as explained in section 3.2) and generates a message $\bar{m}'$, where,

$$\bar{m}' = \begin{cases} CH_{sk^T_{sanit}}(m_{ID}||i||m'_i, r'_i) \text{ for } i \in \{i_1, i_2, ..., i_k\}, \\ \bar{m}_i||i \text{ otherwise} \end{cases}$$

The mutable portion of the message** m **will be replaced with the nodes in the** $Node\ List$ **and signed.

The sanitizing algorithm $SANIT$ takes as input $\bar{m}$, the signature σ on $\bar{m}$, the modified message $\bar{m}'$, the public signing key pk^S_{sign} of the source S, and the private sanitizing key sk^T_{sanit} of the target T to generate the sanitized signature σ', where

$$\sigma' \leftarrow SANIT(\bar{m}, \sigma, \bar{m}'; pk^S_{sign}, sk^T_{sanit}).$$

T then generates a route reply message M_T which consists of the *ids* of source and target, the *request id*, the *Node List* (i.e., the accumulated route obtained from the request), the modified message $\bar{m}'$, and the sanitized signature σ', where

$$M_T = (rrep, \bar{m}', \sigma', S, T, id, (X_1, ..., X_n)).$$

Unlike Ariadne (Hu et al. [2005]), the target does not compute the MAC over the reply because the reply is authenticated using sanitized signature σ'. The reply is then sent back to S on the reverse route included in the route request. When each node X_j in the $Node\ List$ receives the reply, it verifies the sanitized signature σ' of the target, using

$$VERIFY(\bar{m}', \sigma'; pk^S_{sign}, pk^T_{sanit}) \longrightarrow \{TRUE, FALSE\}.$$

If this test returns $FALSE$, X_j discards the reply. If it returns $TRUE$, X_j verifies that the mutable portion of the message has been modified (i.e., the signature has been sanitized) by the target. If not, then X_j discards the reply. If both these tests succeed, X_j unicasts the reply message to X_{j-1} $(1 \leq j \leq n)$, on the reverse route $T = X_{n+1}, X_n, X_{n-1}, ..., X_1, X_0 = S$. Weak transparency of the sanitizable signature gives intermediate nodes the ability to verify whether the signature has been sanitized by T.

When S receives the reply, it checks whether the sanitized signature σ' passes the verification using $VERIFY$; if so, the source accepts the route; otherwise, it discards it. Unlike Ariadne(Hu et al. [2005]), the source does not need to recalculate the corresponding MAC of each intermediate node since the authentication is done through the sanitized signature σ'.

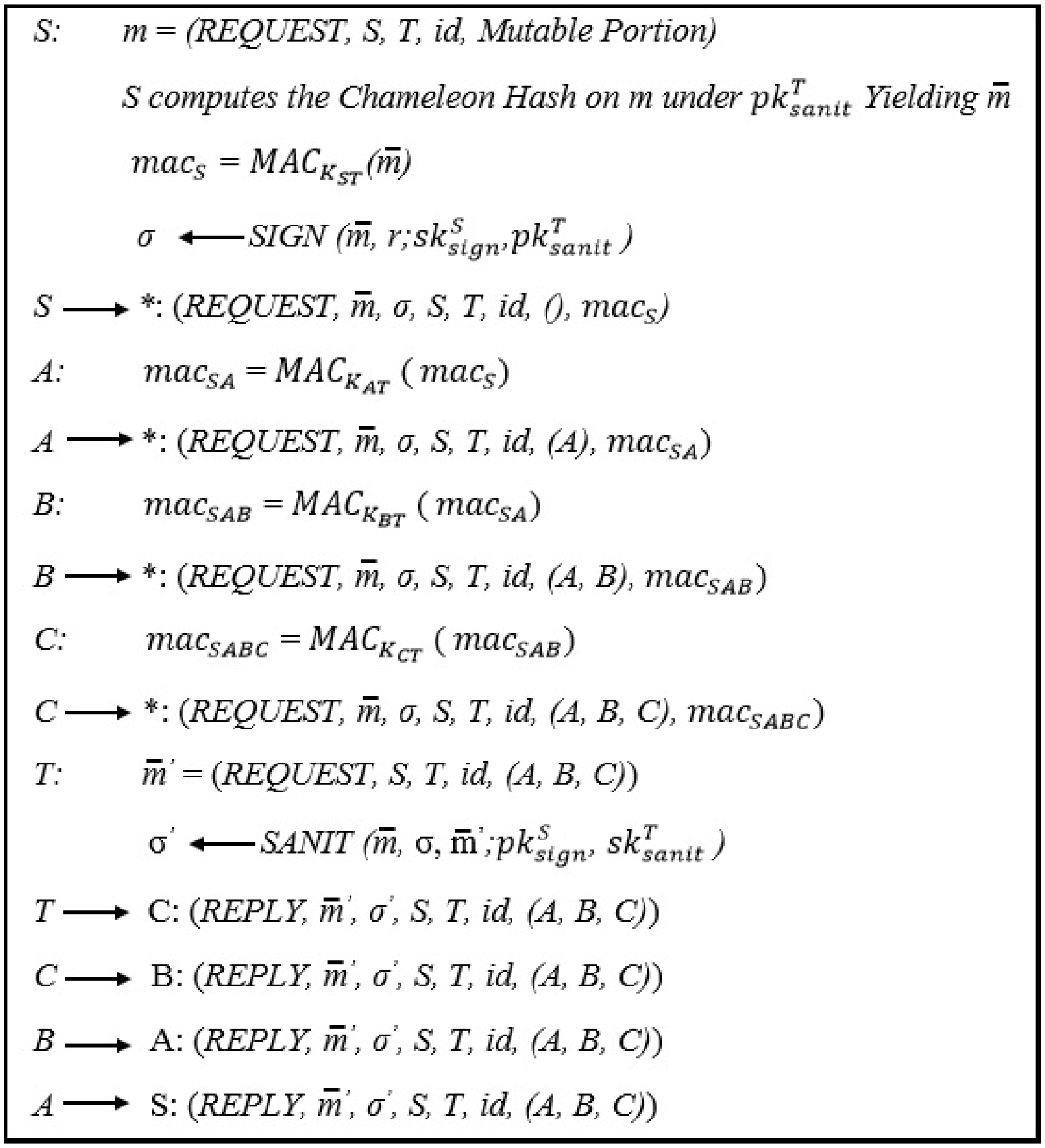
Figure 7 shows the steps involved in route discovery under our protocol when the source tries to discover a route to the target node T and the route request goes through the intermediate nodes *(A,B,C)* before reaching T. Figures 8 and 9 respectively show how route requests and route replies at intermediate nodes are processed under our protocol.

5. ANALYSIS AND DISCUSSION OF THE ATTACKS DETECTED BY THE PROPOSED PROTOCOL

In this section, we discuss and show how our protocol prevents the hidden channel attacks.

5.1 Detecting and Preventing the Hidden Channel Attack on SRP

In SRP, intermediate nodes can easily inject invalid identifiers into route requests and forward them to the target T which in turn verifies these non-existent nodes correctly. This is possible because intermediate nodes neither check the source's MAC nor they append their own MACs to the route request. This kind of attack can be detected by our protocol. Consider the attack explained in section 2.1.1, when the adversary A inserts the arbitrary sequence of identifier λ,



S: $m = (REQUEST, S, T, id, Mutable\ Portion)$

S computes the Chameleon Hash on m under pk^{T}_{sanit} Yielding $\bar{m}$

$mac_S = MAC_{K_{ST}}(\bar{m})$

$\sigma \longleftarrow SIGN\,(\bar{m}, r; sk^{S}_{sign}, pk^{T}_{sanit})$

$S \longrightarrow$ *: $(REQUEST, \bar{m}, \sigma, S, T, id, (), mac_S)$

A: $mac_{SA} = MAC_{K_{AT}}\,(mac_S)$

$A \longrightarrow$ *: $(REQUEST, \bar{m}, \sigma, S, T, id, (A), mac_{SA})$

B: $mac_{SAB} = MAC_{K_{BT}}\,(mac_{SA})$

$B \longrightarrow$ *: $(REQUEST, \bar{m}, \sigma, S, T, id, (A, B), mac_{SAB})$

C: $mac_{SABC} = MAC_{K_{CT}}\,(mac_{SAB})$

$C \longrightarrow$ *: $(REQUEST, \bar{m}, \sigma, S, T, id, (A, B, C), mac_{SABC})$

T: $\bar{m}' = (REQUEST, S, T, id, (A, B, C))$

$\sigma' \longleftarrow SANIT\,(\bar{m}, \sigma, \bar{m}'; pk^{S}_{sign}, sk^{T}_{sanit})$

$T \longrightarrow$ C: $(REPLY, \bar{m}', \sigma', S, T, id, (A, B, C))$

$C \longrightarrow$ B: $(REPLY, \bar{m}', \sigma', S, T, id, (A, B, C))$

$B \longrightarrow$ A: $(REPLY, \bar{m}', \sigma', S, T, id, (A, B, C))$

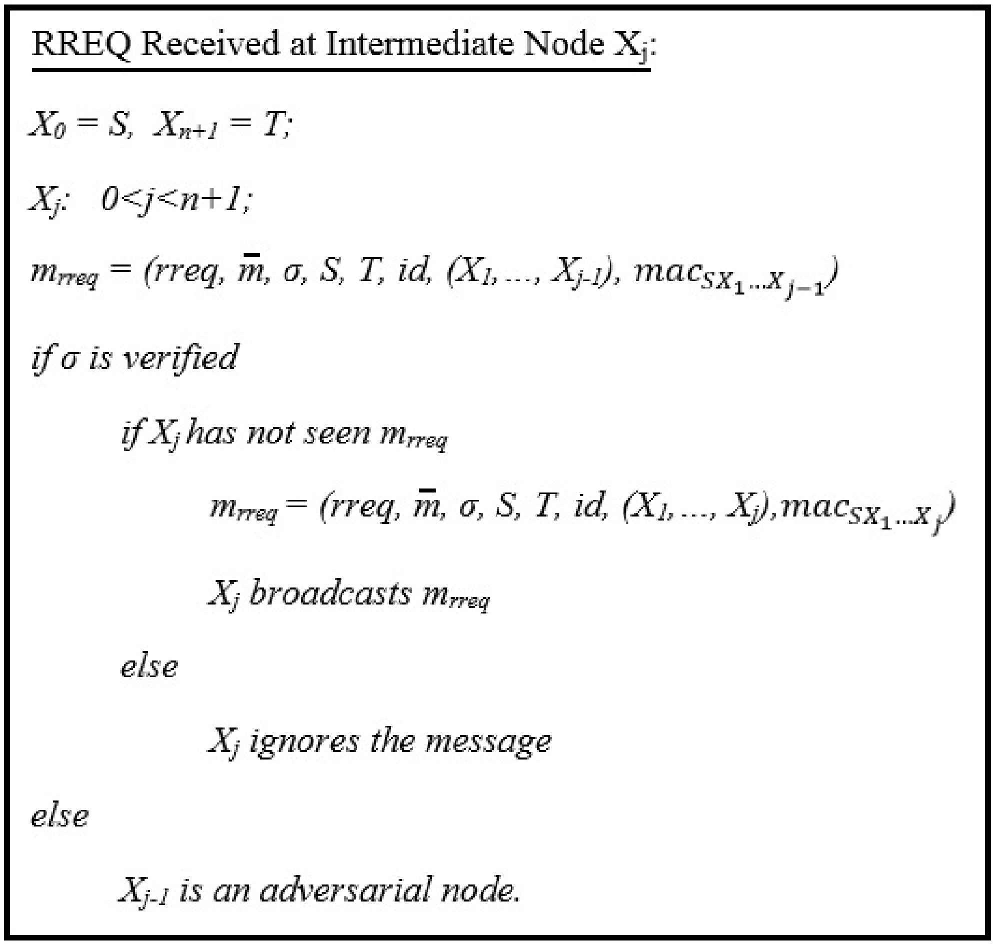
$A \longrightarrow$ S: $(REPLY, \bar{m}', \sigma', S, T, id, (A, B, C))$

An Illustration of Route Discovery Under our Protocol. The source S tries to find a route to the target node T.

under our protocol, the node list cannot pass the verification done by the target. Hence, our protocol prevents such attacks.

5.2 Detecting and Preventing the attack on Ariadne with Signature

In this case, the adversarial node removes a node from the *Node List* (i.e., the route) such that the target can still validate the route (see Section 2.3.1). In Ariadne with Signature (see Section 2.3), each node appends the signature generated over the fields of the received route request. This gives ability to adversarial nodes (e.g., node A in Figure 2) to reconstruct the signatures when they receive multiple requests before rebroadcasting any of them. This signature technique leads to information leakage (attack) like the one presented by Buttyán and Vajda [2004], as explained in section 2.3.1. However, this attack is prevented by our protocol. When iterating MAC is used, the adversarial node cannot reconstruct the MAC using multiple requests. For



```
RREQ Received at Intermediate Node $X_j$:
$X_0 = S,\ X_{n+1} = T;$
$X_j$:   $0<j<n+1;$
$m_{rreq} = (rreq, \bar{m}, \sigma, S, T, id, (X_1, \ldots, X_{j-1}), mac_{SX_1 \ldots X_{j-1}})$
if $\sigma$ is verified
    if $X_j$ has not seen $m_{rreq}$
        $m_{rreq} = (rreq, \bar{m}, \sigma, S, T, id, (X_1, \ldots, X_j), mac_{SX_1 \ldots X_j})$
        $X_j$ broadcasts $m_{rreq}$
    else
        $X_j$ ignores the message
else
    $X_{j-1}$ is an adversarial node.
```

Handling of Route Request by an Intermediate Node X_j.

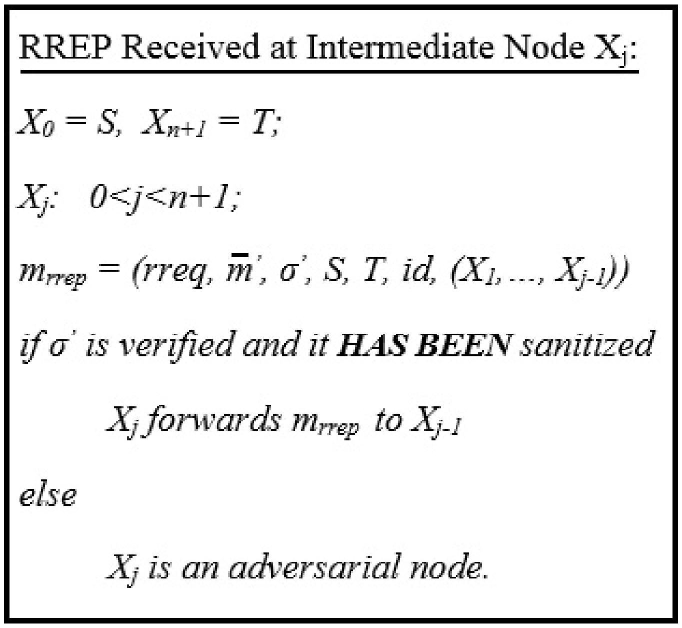
example, in Figure 2, where S initiates a route request to discover a route to T, A receives the following two messages from nodes V and X respectively under our protocol

$$msg_1 = (rreq, \bar{m}, \sigma, S, T, id, (Q, V), mac_{SQV}), \text{ and}$$
$$msg_2 = (rreq, \bar{m}, \sigma, S, T, id, (Q, V, W, X), mac_{SQVWX}).$$

Now, from message msg_2 A knows that V and W are neighbors, and A needs to create a message msg_3, where

$$msg_3 = (rreq, \bar{m}, \sigma, S, T, id, (Q, V, W, A), mac_{SQVWA}).$$

A needs to broadcast this message so that the target can validate all the intermediate nodes with their corresponding keys. However, in order to do that (i.e., to remove X from the route), A needs to get the iterated MAC calculated by W (i.e., mac_{SQVW}) which is protected by W's key. Hence A cannot remove X unless it knows the secrete key of W to calculate W's MAC (i.e., mac_{QVW}) and as a result, msg_3 cannot be created. Thus, our protocol prevents hidden channel attacks.



RREP Received at Intermediate Node X_j:

```
$X_0 = S,\ X_{n+1} = T;$
$X_j$:  $0<j<n+1;$
$m_{rrep} = (rreq, \bar{m}', \sigma', S, T, id, (X_1, \ldots, X_{j-1}))$
if $\sigma'$ is verified and it HAS BEEN sanitized
    $X_j$ forwards $m_{rrep}$ to $X_{j-1}$
else
    $X_j$ is an adversarial node.
```

Handling of Route Reply by an Intermediate Node X_j

5.3 Detecting and Preventing the Attack on Basic Ariadne with MAC

Adversarial nodes in this scheme collaborate to remove a node or a sequence of nodes located between two of the adversarial nodes as explained in Section 2.4.1. Intermediate nodes in Ariadne with MAC append their calculated MAC values to a list of MACs giving adversaries enough space to shorten the real route between source and target nodes. Consider the attack presented in section 2.4.1, when the request arrives at the second adversarial node Y (see Figure 3), it drops the sequence (B, M, L, C) from the request.

Now, let's assume that the adversarial node can successfully remove the sequence mentioned above from the route request and the request arrives at T. Now, under our protocol, this attack is detected as follows. The target T puts the node list in the sanitizable position of the sanitizable signature which in turn can be verified by each intermediate node. The accumulated route that is sanitized in this example would be (A, Y, D). Y cannot re-insert the hidden nodes (i.e., (B, M, L, C)) into the route reply received from D because each of these nodes can verify the sanitized signature which was computed by T on the received path (i.e., node list). If Y does so, the first node C in the node list will detect Y as an adversarial node because the sanitized signature will not be successfully verified. This prevents the reply from arriving at the source and hence, non-plausible routes are not created. Thus, our protocol prevents this type of hidden channel attacks.

5.4 Detecting and Preventing the attack on the Optimized Version of Ariadne with Iterated MAC

The iterating MAC calculated at each intermediate node is considered more efficient than the other flavors of Ariadne to prevent attacks. However, this optimized version is also susceptible to hidden channel attacks as discovered by Burmester and de Medeiros [2009] and discussed in section 2.5.1. This is because under this version of Ariadne, intermediate nodes check neither iterated MACs calculated by intermediate nodes nor the MAC computed by the target T.

In our protocol every node verifies that the signature is sanitized, hence, when a route reply arrives at intermediate nodes, it will be forwarded only if the verification of the sanitized signature succeeds.

Now, let's explain how our protocol prevents this type of hidden channel attack. Consider the hidden channel attack against Ariadne presented in section 2.5.1; we notice that the adversarial node Y (see Figure 4) needs the iterating MAC (mac_{SAX}) that was broadcast by the other adversarial node X. But since there is an extra node, B, between X and Y, X needs to impersonate Y and unicast a fake reply to B such that when B forwards the reply to X, Y can intercept the reply and extract the MAC generated by X (i.e., mac_{SAX}).

Now when X creates a fake route reply in the name of Y and unicasts it to B (Burmester and de Medeiros [2009], see Section 2.5.1), B checks the signature and finds it is not sanitized because T has not received the route request yet, hence B detects that Y is an adversarial node. Thus, our protocol prevents hidden channel attacks present in optimized Ariadne with iterated MAC.

Then, after the reply is sanitized by T, the route reply cannot be tampered with by an intermediate node. The source or the next upstream/downstream intermediate node to the adversarial node can detect the modification, hence the route reply arrives intact at the source S.

It is noteworthy to mention that we use one signature that is signed by the source S and sanitized by the target T. This signature is verified by the intermediate nodes in the request phase as well as in the reply phase. This important feature makes the proposed routing protocol securely propagate route requests towards the target node (downstream flow) and also securely forward route replies towards source node (upstream flow) at a cost of one signature which is more cost effective than other schemes in which each node uses its own individual signature.

5.5 Detecting and Preventing the attack on endairA

The *endairA* protocol secures route replies by digitally signing but does not secure route requests. Hence adversarial nodes need a strategy to hide control information while route replies are forwarded to source nodes. Therefore, the attack found by Burmester and de Medeiros [2009] on *endariA* presented in Section 2.6.1 was based on this strategy. The adversarial node Y (Figure 6) needs to send the signature list (i.e., (sig_T, sig_D, sig_Y)) to the other adversarial node X so that the source S accepts the reply. It is assumed that there has been a route discovery initiated by node D towards the target A prior to the route discovery from S to T. Y uses this previous route discovery to hide the signature list in the id of the route request so that X can reconstruct the required information (i.e., (sig_T, sig_D, sig_Y)) from the modified id. The main reason for this attack is that route requests are not secured.

This attack is prevented by our protocol because the main part m (Section 4.2.1) of route requests in our protocol is authenticated, where

$$m = (id, m_{ID}, S, T, sanitizable - portion).$$

As explained in Section 4.2.1, m, which is an intrinsic part of the request, is signed by the source S using the sanitizable signature. Each intermediate node verifies this sanitizable signature, which is part of the request, before forwarding the request.

Now, to detect the attack on endairA, we go back to the scenario mentioned in Section 2.6.1 and briefly explained above. When the second adversarial node Y modifies id' of the request into some other identifier id'' to contain the signature list, the main portion of the route request gets modified. This modification yields a signature different from the one sent the route request; hence, when the neighbor nodes of Y (i.e., intermediate nodes) receive this request, they discard it because the signature cannot be verified. Therefore, the modified request cannot travel any further beyond the neighbors of Y which means Y cannot communicate with X using invalid request information. As a result, this verification prevents such attacks which in turn means that our protocol prevents hidden channel attack present in endairA.

6. CONCLUSION

In this paper, we discussed how some of the secure routing protocols proposed for MANETs are in fact not secure, and presented a novel, secure routing protocol for MANETs. The proposed protocol relies on sanitizable signature, a scheme that allows other parties to sanitize the signature so that the original signer as well as other intermediate nodes can validate the sanitized signature. It prevents hidden channel attacks for MANETs. The security mechanism we designed is general and can be applied to any source routing protocol. A comprehensive analysis of the hidden channel attacks on SRP, Ariadne with its three flavors, and endairA is provided and we showed how our protocol detects and prevents these hidden channel attacks.

References

Abdelaziz, A. K., Nafaa, M., and Salim, G. 2013. Survey of routing attacks and countermeasures in mobile ad hoc networks. In *Proceedings of 2013 IEEE International Conference on Computer Modelling and Simulation (UKSim).*

Ács, G., Buttyán, L., and Vajda, I. 2006. Provably secure on-demand source routing in mobile ad hoc networks. *IEEE Transactions on Mobile Computing 5,* 11, 1533–1546.

Arulkumaran, G. and Gnanamurthy, R. K. 2019. Fuzzy trust approach for detecting black hole attack in mobile adhoc network. *Mobile Networks and Applications 24,* 2, 386–393.

Ateniese, G., Chou, D. H., Medeiros, B., and Tsudik, G. 2005. Sanitizable signatures. In *Proceedings of 10th European Symposium on Research in Computer Security.* Springer Berlin Heidelberg, 159–177.

Ateniese, G. and Medeiros, B. 2004. *Security in Communication Networks: International Conference, SCN.* Springer Berlin Heidelberg, Chapter On the Key Exposure Problem in Chameleon Hashes, 165–179.

Burmester, M. and de Medeiros, B. 2009. On the security of route discovery in MANETs. *IEEE Transactions on Mobile Computing 8,* 9 (Sept), 1180–1188.

Buttyán, L. and Vajda, I. 2004. Towards provable security for ad hoc routing protocols. In *Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks.* ACM, New York, NY, USA.

Giruka, V. C. and Singhal, M. 2007. A self-healing on-demand geographic path routing protocol for mobile ad-hoc networks. *Ad Hoc Networks 5,* 7 (Sept.), 1113–1128.

Hsu, C. C. and Lei, C. L. 2009. A geographic scheme with location update for ad hoc routing. In *Proceedings of Fourth International Conference on Systems and Networks Communications, ICSNC.*

Hu, Y., Perrig, A., and Johnson, D. B. 2003. Efficient security mechanisms for routing protocols. In *Proceedings of NDSS.*

Hu, Y.-C., Perrig, A., and Johnson, D. B. 2005. Ariadne: A secure on-demand routing protocol for ad hoc networks. *Wireless networks 11,* 1-2, 21–38.

Hurley-Smith, D., Wetherall, J., and Adekunle, A. 2017. SUPERMAN: security using pre-existing routing for mobile ad hoc networks. *IEEE Transactions on Mobile Computing 16,* 10, 2927–2940.

Jain, S., Shastri, A., and Chaurasia, B. K. 2013. Analysis and feasibility of reactive routing protocols with malicious nodes in manets. In *Proceedings of International Conference on Communication Systems and Network Technologies (CSNT).*

Johnson, D. B. and Maltz, D. A. 1996. *Mobile Computing.* Springer US, Chapter Dynamic Source Routing in Ad Hoc Wireless Networks, 153–181.

Karp, B. and Kung, H. T. 2000. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking.*

Kavitha, K., Selvakumar, K., Nithya, T., and Sathyabama, S. 2013. Zone based multicast routing protocol for mobile ad hoc network. In *Proceedings of International Conference on*

Emerging Trends in VLSI, Embedded System, Nano Electronics and Telecommunication System (ICEVENT).

Kojima, H., Yanai, N., and Cruz, J. P. 2019. ISDSR+: improving the security and availability of secure routing protocol. *IEEE Access 7*, 74849–74868.

Krawczyk, H., Bellare, M., and Canetti, R. 1997. HMAC: Keyed-hashing for message authentication. RFC 2104, RFC Editor. February.

Li, H. and Singhal, M. 2005. An anchor-based routing protocol with cell id management system for ad hoc networks. In *Proceedings of International Conference on Computer Communications and Networks.*

Li, T., Ma, J., and Sun, C. 2019. SRDPV: Secure route discovery and privacy-preserving verification in manets. *Wireless Networks 25*, 1731–1747.

Papadimitratos, P. and Haas, Z. 2002. Secure routing for mobile ad hoc networks. *Mobile Computing and Communications Review 1,* 2, 27–31.

Perkins, C. E. and Bhagwat, P. 1994. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. *SIGCOMM Computing Commununications Review 24,* 4 (Oct.), 234–244.

Perkins, C. E. and Royer, E. M. 1999. Ad-hoc on-demand distance vector routing. In *Proceedings of Second IEEE Workshop on Mobile Computing Systems and Applications (WMCSA).*

Perrig, A., Canetti, R., Tygar, J., and Song, D. 2000. Efficient authentication and signing of multicast streams over lossy channels. In *Proceedings of IEEE Symposium on Security and Privacy.*

Rivest, R., Shamir, A., and Adleman, L. 1978. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM 21*, 120–126.

Salehi, M. and Boukerche, A. 2019. Secure opportunistic routing protocols: methods, models, and classification. *Wireless Networks 25,* 2, 559–571.

Shen, H. and Zhao, L. 2013. ALERT: An anonymous location-based efficient routing protocol in MANETs. *IEEE Transactions on Mobile Computing 12,* 6 (June), 1079–1093.

Talooki, V. N., Marques, H., and Rodriguez, J. 2013. Energy efficient dynamic MANET on-demand (E2DYMO) routing protocol. In *Proceedings of International Symposium and Workshops on a World of Wireless, Mobile and Multimedia Networks (WoWMoM).*

Xia, H., shun Zhang, S., Li, Y., kuan Pan, Z., Peng, X., and Cheng, X. 2019. An attack-resistant trust inference model for securing routing in vehicular ad hoc networks. *IEEE Transactions on Vehicular Technology 68,* 7, 7108–7120.

Xu, Y., Liu, J., Shen, Y., Jiang, X., and Shiratorie, N. 2017. Physical layer security-aware routing and performance tradeoffs in ad hoc networks. *Computer Networks 123,* 4, 77–87.

Yih-Chun, H. and Perrig, A. 2004. A survey of secure wireless ad hoc routing. *IEEE Security Privacy 2,* 3 (May), 28–39.

Dr. Baban A. Mahmood is currently the Chair of The Department of Networking, College of Computer Science and Information Technology at University of Kikuk, Kikuk, Iraq. He received his Ph.D in Computer Science from University of Kentucky, Lexington, Kentucky, USA. He received a B.Sc, degree in Computer and Software engineering from University of Al-Mustansryah, Baghdad, Iraq, in 2003 and an M.Sc. degree in Computer Science from University of Sulaimaniya, Kurdistan, Iraq, in 2009. He also received an M.Sc., degree in Computer Science from University of Kentucky, Lexington, Kentucky, USA, in 2015. He published his research work in the area of routing protocols in mobile ad hoc networks.

Dr. D. Manivannan is currently an Associate Professor in the Computer Science Department at University of Kentucky, Lexington, Kentucky, USA. He received a B.Sc degree in Mathematics from University of Madras, Madras, India. He received an M.S in Mathematics, an M.S in Computer Science and a PhD in computer and information science from The Ohio State University, Columbus, Ohio, USA, in the years 1992, 1993 and 1997 respectively. He published his research work in the following areas: fault-tolerance and synchronization in distributed systems, security and fault-tolerance in cloud computing systems, routing in wormhole networks, routing in mobile ad hoc networks and vehicular ad hoc networks, vehicular clouds, channel allocation in cellular networks and wireless personal area networks. Dr. Manivannan has published more than 80 articles in refereed International Journals (a vast majority of which were published by IEEE, ACM, Elsevier, and Springer) and Proceedings of International Conferences.
Dr. Manivannan served as an Associate Editor of IEEE Transactions on Parallel and Distributed Systems, IEEE Communications Magazine and Wireless Personal Communications journal. He served as Program co-chair of three International Conferences in the areas of reliable distributed systems and wireless networks and served as program committee member for over 40 International Conferences. He is on the Editorial Board of Information Sciences journal. He served as reviewer for more than 35 International Journals published by ACM, IEEE, Elsevier, Springer, Oxford University Press, Taylor and Francis and others. He also served on several proposal review panels of US National Science Foundation and as external tenure reviewer for other Universities. Dr. Manivannan's research has been funded by grants from the US National Science Foundation and the US Department of Treasury.
Dr. Manivannan is a recipient of the Faculty Early Career Development award (CAREER award) from US National Science Foundation. He is a senior member of the IEEE and ACM.