ASRPAKE: An Anonymous Secure Routing Protocol with Authenticated Key Exchange for Wireless Ad Hoc Networks

Xiaodong Lin*, Rongxing Lu†, Haojin Zhu*, Pin-Han Ho*, Xuemin (Sherman) Shen* and Zhenfu Cao†
*Department of Electrical and Computer Engineering
University of Waterloo, Waterloo, Ontario, Canada
Email: {xdlin,h9zhu,pinhan,xshen}@bbcr.uwaterloo.ca
†Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai, P. R. China
Email: {rxlu,zfcao}@cs.sjtu.edu.cn

Abstract— **In this paper, we present a novel anonymous secure routing protocol for *mobile ad hoc networks* (MANETs). The proposed protocol not only provides anonymity from all the intermediate nodes, but also integrates the authenticated key exchange mechanisms into the routing algorithm design. Furthermore, a new attack on anonymous services, called *snare attack*, is introduced, where a compromised node lures a *very important node* (VIN) into communicating with him and traces back to the VIN by following the route path. An adversary can then snare the VIN and launch *Decapitation Strike* on the VIN. Finally, we present a novel DECOY mechanism as a countermeasure to enhance anonymity of VINs and defeat *snare attack*.**

Keywords – **Wireless ad hoc network, anonymous, security, routing protocol, authenticated key exchange, snare attack.**

I. Introduction

Mobile ad hoc network (MANET) is composed of a set of autonomous wireless nodes. Because of the nature of rapid deployment and absence of fixed network infrastructure, MANET plays a major role in establishing communication for diverse situations such as emergency and natural disaster relief, military conflicts, and some commercial applications. In a MANET, each node is usually as simple as a laptop or a personal communication device, and typically acts as a role of both router and host at the same time. When these nodes perform routing and packet forwarding, they may not have sufficient protection from malicious attacks; instead, the network security has to be maintained through the robustness and security of signaling protocols. Due to limited communication range of these mobile nodes and lack of centralized monitoring and management, establishing an anonymous secure route in a MANET is not as a trivial work as that in a wired network. In this aspect, ensuring security and anonymity in a MANET is more than critical to its overall success. Recent research efforts have been appeared in [1], [2], [3], [4].

In this paper, we present a novel anonymous secure routing protocol with authenticated key exchange (ASRPAKE) for MANETs. The major advantages of our ASRPAKE protocol lie in the following two aspects: 1) it provides anonymity to the route from the source to the destination; 2) it integrates a suite of interoperative authenticated key exchange mechanisms into the routing algorithm design. Further, we introduce a new attack on anonymous services, called *snare attack*, where a compromised node lures a *very important node* (VIN) to communicate and traces back to the VIN by following the route path of the communication. Afterwards, the adversary can snare the VIN and launch *Decapitation Strike* on the VIN. Finally, we present a novel DECOY mechanism as a countermeasure to enhance the anonymity of the VINs and defeat the snare attack.

The rest of this paper is organized as follows. In section II, we propose an efficient ring signature scheme based on *Elliptic Curve Cryptosystem* (ECC) [5] to achieve anonymous authentication key agreement among mobile nodes in the network. In section III, we present our ASRPAKE protocol. In section IV, we introduce a new attack on anonymous services, called *snare attack*, and then present a DECOY mechanism as a countermeasure to enhance anonymity. The anonymity and security of the proposed protocol is analyzed and discussed in section V. Finally, Section VI concludes the paper.

II. Anonymous Authenticated Key Agreement Protocol

In this section, we propose an efficient ring signature scheme based on ECC to achieve anonymous authenticated key agreement among mobile nodes in the network. Our ring signature scheme is an enhancement of the provably secure ring signature scheme, which provides unconditional anonymity [6]. Ring signature is a group-oriented signature providing anonymity for signers where any user from the group can sign a message on behalf of a set of members (including himself) such that a verifier can be convinced that the message has been signed by one of the group member without knowing which the actual signer is. Ring signature can successfully provide anonymity to the group of members, however, the signature overhead for the traditional ring signature schemes grows with the group size [6], which makes it infeasible in MANETs due to the constraints on the computation power, memory size, and battery capacity of each mobile node [7].

Let $p > 3$ be a large prime. Two field elements $a, b \in \mathbb{Z}_p$ are chosen such that $4a^3 + 27b^2 \neq 0 (mod\ p)$ in order to define the equation of a non-supersingular elliptic curve $\mathbf{E} : y^2 = x^3 + ax + b(mod\ p)$ over $\mathbb{Z}_p$. We define $\mathbf{E}(\mathbb{Z}_p)$ as a group for the set of solutions $(x, y) \in \mathbb{Z}_p \times \mathbb{Z}_p$ to the congruence $y^2 = x^3 + ax + b(mod\ p)$ together with a special point $\mathcal{O}$ called the point at infinity. With $\mathbf{E}(\mathbb{Z}_p)$ being defined, a generator point $P = (x_p, y_p)$ is chosen such that its order is a large prime number q over $\mathbf{E}(\mathbb{Z}_p)$, where $P \neq \mathcal{O}$. In such a way, a subgroup $\mathbf{G}$ of the elliptic curve group $\mathbf{E}(\mathbb{Z}_p)$ with order q is constructed.

By considering a set of potential signers $\mathcal{U} = \{\mathcal{U}_1, \mathcal{U}_2, ..., \mathcal{U}_n\}$, each user $\mathcal{U}_i$ has a private key $x_i \in \mathbb{Z}_q^*$ and the corresponding public key $Y_i = x_i P$. Choose a secure hash function $H : \mathbf{G} \times \mathbf{G} \rightarrow \mathbb{Z}_q^*$.

The proposed ring signature scheme consists of the execution of the two following algorithms:

- **Ring-sign:** Choose a random number $x \in \mathbb{Z}_q^*$ and compute xP. To sign xP on behalf of the group $\mathcal{N}$ from the ring $\mathcal{U} = \{\mathcal{U}_1, \mathcal{U}_2, ..., \mathcal{U}_n\}$ where $|\mathcal{N}| = m \leq n$, a signer $\mathcal{U}_u \in \mathcal{N}$ carries out as follows:
  1. For all $i \in \{1, ..., m\}$ and $\mathcal{U}_i \neq \mathcal{U}_u$, $\mathcal{U}_u$ randomly chooses $a_i \in \mathbb{Z}_q^*$ and for which the a_i are pairwise different. Compute
  $$R_i = a_i P \quad (i \neq u)$$
  2. Choose a random number $a \in \mathbb{Z}_q^*$.
  3. Compute R_u, where
  $$R_u = aP - \sum_{i=1, i \neq u}^{m} H(xP, R_i) Y_i$$
  If $R_u = \mathcal{O}$ or $R_u = R_i$ for some $i \neq u$, then go to step 2.
  4. Compute σ, where
  $$\sigma = a + \sum_{i=1, i \neq u}^{m} a_i + x_u H(xP, R_u) \ mod q$$
  5. Then, the signature of xP made by the group $\mathcal{N}$ from the ring $\mathcal{U} = \{\mathcal{U}_1, \mathcal{U}_2, ..., \mathcal{U}_n\}$ to be $(R_1, ..., R_m, Y_1, ..., Y_m, \sigma)$.

- **Ring-verify:** Anyone can verify the signature by the followings:
  1. Compute h_i, where
  $$h_i = H(xP, R_i) \quad for \ all \ 1 \leq i \leq m$$
  2. Check the equation
  $$\sigma P = \sum_{i=1}^{m} (R_i + h_i Y_i)$$

The correction is shown as follows:

$$\begin{aligned} \sum_{i=1}^{m} (R_i + h_i Y_i) &= R_u + h_u Y_u + \sum_{i=1, i \neq u}^{m} (R_i + h_i Y_i) \\ &= aP + h_u Y_u + \sum_{i=1, i \neq u}^{m} R_i \\ &= \sigma P \end{aligned}$$

It is worth to note that in the proposed ring signature scheme, only a subset of group members in the ring have been chosen to generate the ring signature, which is different from other traditional ring signature schemes. Nevertheless, the anonymity strength of the proposed scheme varies based on m, the size of the chosen signing group. The larger the size of the signing group is, the more anonymous the proposed ring signature scheme can be. However, a large size of a signing group which incurs large signature overhead may cause a serious concern when digital signature schemes are applied in a real environment, especially in wireless ad hoc environment.

Finally, an anonymous authenticated key agreement protocol between Alice and Bob based on the proposed ring signature scheme can be established as follows:

Alice		Bob
xP	$\xrightarrow{xP, R_1, ..., R_m, Y_1, ..., Y_m, \sigma}$	
	$\xleftarrow{yP, R_1', ..., R_m', Y_1', ..., Y_m', \sigma'}$	yP
$k = x(yP)$		$k = y(xP)$
Anonymous authenticated key agreement		

Assume that both Alice and Bob are from the ring $\mathcal{U} = \{\mathcal{U}_1, \mathcal{U}_2, ..., \mathcal{U}_n\}$. They want to anonymously authenticate each other, where both Alice and Bob know that they are talking to an authentic peer in the ring without knowing the real identity of their peer. In addition, Alice and Bob should possess a new session key at the end of protocol. To achieve the above design objectives, the following anonymous authentication mechanisms are devised. Both Alice and Bob choose a random number $x \in \mathbb{Z}_q^*$ and $y \in \mathbb{Z}_q^*$, and compute xP and yP, respectively. Afterwards, each of them randomly chooses m members from the ring including himself/herself and signs xP and yP by using the above ring signature scheme, respectively. The derived signature is then sent to the other. Finally, each of them verifies the received signature from his/her peer. If the authentication succeeds, they can compute the session key respectively as follows:

$$\begin{aligned} k_{ab} &= x(yP) \quad by \ Alice \\ &= y(xP) \quad by \ Bob \end{aligned}$$

III. Anonymous Secure Routing Protocol

A. System Formulation

- A bidirectional link between two mobile nodes within the transmission range can be established in the MANET.
- Each node maintains two tables for facilitating the anonymous routing mechanisms: one is a local neighborhood table, whose format of each entry is shown as follows:

Neighbor Address	Session Key	Life Time

where the first field records its neighbor node's address; the second field records its session key between itself and the corresponding neighbor, which is used to ensure confidentiality and integrity of the transmitted messages; and the third field records a timer which controls how

long the corresponding neighbor is active. If the timer hits 0, the entry will be removed.
The other is the local route table, and the format of each entry in the route table is shown as follows:

rt_sequence	Dest_ID	Ancenstor	Sucessor	Life Time

where the first field represents a unique route, which is a hash value of the source address and its source sequence number; the second field records the identity of the destination for the source or the identity of the source for the destination. However, it is not applicable for any intermediate nodes; the third field records its upstream node's address; the fourth field records its downstream node's address; and the last field records a timer which controls how long the route is valid. If the timer hits 0, the entry will be removed.

- In the wireless ad hoc networks, some intermediate nodes along the route may try to violate the anonymity property, however we can assume that it is with negligible probability that all the intermediate nodes are in collusion.

B. Description of Protocol

The proposed anonymous secure routing protocol consists of five phases: the *key pre-distribution phase*, the *neighborhood discovery phase*, the *route discovery phase*, the *route reverse phase*, and the *data forwarding phase*. The notations are listed in Table I.

TABLE I
NOTATIONS

Notation	Description of the Notation
S, **D**:	The source and the destination node
I_i:	the i-th intermediate node on the route from **S** to **D**
ID_S:	The identity of source **S**
ID_D:	The identity of destination **D**
ID_i:	The identity of the i-th intermediate node
(ID_N, S_{ID_N})	The Identity-based public/private key pair of the node N with the identity as ID_N
N_Addr	The address of the node N with the identity as ID_N
p:	A secure large prime number
g:	A generator with order $(p-1)$ in $GF(p)$
SK_{SD}:	The established session key between **S** and **D**
$E_{ID_N}(m)$:	Encryption of message m by using any implicit Identity-Based Encryption scheme with the public key of the node N.
$e_k(m)$:	Encryption of message m by using any implicit secure symmetric encryption algorithm, i.e. DES [12], under the key of k
$H(.)$:	A secure one-way hash function, i.e. SHA-1 [12]
$MAC_k(m)$:	The keyed-hash message authentication code (HMAC) on the message m by applying any implicit secure HMAC function [15] under the key of k
$\mathtt{T}_S, \mathtt{T}_D, \mathtt{T}_i$:	The timer of **S**, **D** and I_i
$\|\|$	stands for message concatenation operation, which appends several messages together in a special format.

1) The Key Pre-Distribution Phase: We assume that an offline *security manager* (SM) exists for identity check and private key pre-distribution. Prior to the network deployment, the SM sets up system parameters as follows:

a. Let G be a generator of $\mathbb{G}_1$, where $\mathbb{G}_1$ is an additive group of prime order q. Let $\mathbb{G}_2$ be a multiplicative group with the same order as $\mathbb{G}_1$ and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be the bilinear pairing. Define one secure hash functions H_1, where $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$;

b. The SM randomly selects $s \in \mathbb{Z}_q^*$ as its master key, and computes $P_{pub} = sG$ as its public key;

c. The SM calculates for each node A an Identity-based public/private key pair (ID_A, S_{ID_A}), where ID_A is the identity of node A, $Q_{ID_A} = H_1(ID_A)$ is node A' s public key, $S_{ID_A} = sQ_{ID_A}$ is node A' s private key.

d. Each node A is preloaded with the public parameters $< \mathbb{G}_1, \mathbb{G}_2, \hat{e}, q, G, P_{pub}, H_1 >$ and its private key S_{ID_A}.

2) Neighborhood Discovery Phase: Whenever a node is brought up, it has to be authenticated by its neighbors through broadcasting a local authentication message without revealing its real identity. The authentication procedure between **A** and N_1 is demonstrated as follows.

Step 1: $A \rightarrow * : n_1, xP$

Step 2: $N_1 \rightarrow A : n_2, yP, R_1^{'}, ..., R_m^{'}, Y_1^{'}, ..., Y_m^{'}, \sigma^{'}$, $MAC_{sk}(N_1\_Addr||n_1||n_2)$

Step 3: $A \rightarrow N_1 : R_1, ..., R_m, Y_1, ..., Y_m, \sigma$, $MAC_{sk}(A\_Addr||n_1||n_2)$

where, $sk = xyP$, n_1 is a random nonce chosen by node **A**, and n_2 is a random nonce chosen by node N_1.

Suppose node **A** enters the network and moves into an area where node N_1 is situated within Node **A**'s signal transmission range. Node **A** chooses a random number $x \in \mathbb{Z}_q^*$, and computes xP. Afterwards, it broadcasts a local authentication message containing a random nonce n_1 and xP. Each neighbor node of Node **A** receives this message. Upon receiving the message, Node N_1 randomly chooses a number $y \in \mathbb{Z}_q^*$ and calculates yP. It signs yP by using the previously proposed ring signature scheme. It computes the shared secret key $sk = y(xP)$. Afterwards, it unicasts a response containing a random nonce n_2, yP, the ring signature on yP, and a message authentication code derived upon Node N_1's address, n_1 and n_2 under the key of sk back to Node **A**. When this message is received, Node **A** verifies the signature on yP. If verification succeeds, Node **A** computes the shared secret key $sk = x(yP)$. Further, it verifies $MAC_{sk}(N_1\_Addr||n_1||n_2)$. If the verification succeeds, it creates a message authentication code of Node **A**'s address, n_1 and n_2, and transmits it to the Node N_1 with the ring signature on xP. At the end, Node N_1 verifies the signature on xP. If the authentication succeeds, Node N_1 verifies whether or not the received nonce message can fit into the calculation of a message authentication code through a concatenated message of Node **A**'s address, n_1 and n_2 under the key sk. If so, the local authentication and key agreement procedure completes successfully; otherwise, Node N_1 rejects, which means that the authentication process fails.

After a successful authentication, it can be assured that both node **A** and N_1 are talking to an authentic peer in the network without any knowledge on the real identity of its peer. Also, a secret key shared between any pair of neighbor nodes is generated. In the end, Node **A** and N_1 insert the entry $|N_1\_Addr|xyP|T_A|$ and $|A\_Addr|xyP|T_{N_1}|$ to its neighborhood table, respectively.

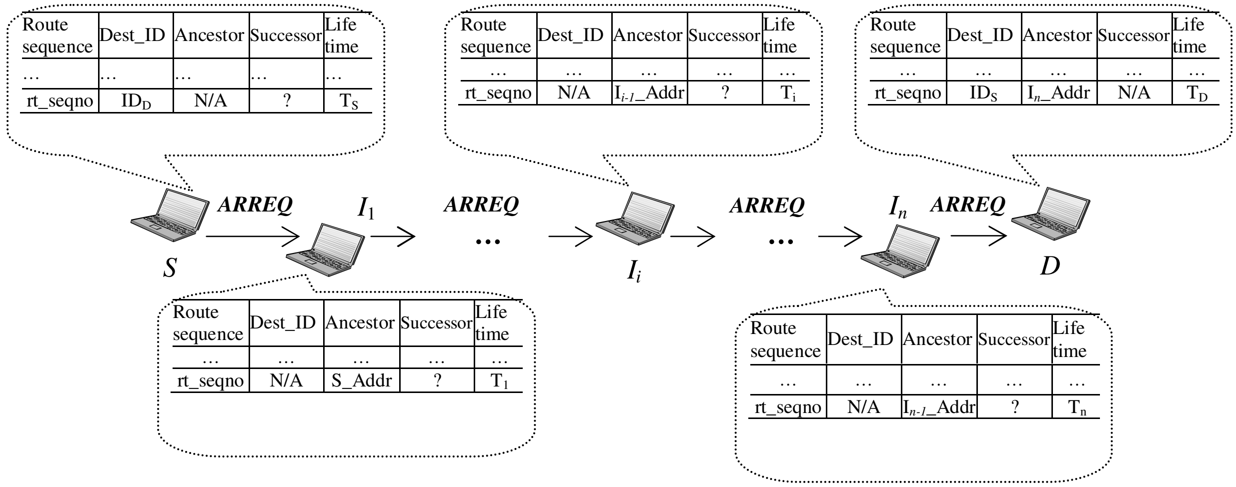

Fig. 1. Route Tables During Route Discovery Phase

With the above arrangement, any node could securely transmit data to all the neighbor nodes. To keep track of each neighbor due to node mobility, a node could simply listen to a local authentication message broadcast by each node at the preset interval. However, the broadcasting mechanism is time- and resource-consuming since a node has to perform the entire authentication process whenever it receives a local authentication message. In order to make the neighbor discovery procedure as efficient as possible, upon receiving a local authentication message, the sender can be alternatively checked against the neighbor table first by the receiver before it further goes through the authentication procedure. If the sender can be found in the table, the message will be discarded by the receiver without going through the full authentication procedure. In this case, the receiver updates the timer of the corresponding entry in the table. If the sender cannot be found in the table, the regular authentication procedure will be initiated, and a new entry corresponding to the sender will be created in the neighbor table of the receiver when the authentication procedure passes.

3) Route Discovery Phase: ASRAPKE uses a broadcast route discovery mechanism as in AODV [13] and DSR [14]. Whenever the source **S** sends confidential data to the destination **D** without either an available route path or a shared session key with **D**, it will first establish a route and a session key shared with **D** in this phase.

Step 1. **S** generates its unique sequence number `src_seq#` for this route path. The sequence number uniquely identifies the particular *anonymous route request* (**ARREQ**) message when taken in conjunction with the source address *S_Addr*. **S** calculates $H(S\_Addr||src\_seq\#)$, denoted by rt_seqno, and selects a random number $a \in [1, p-1]$ to compute g^a and $H(g^a||K_{SD}||0)$, where $K_{SD} = \hat{e}(H(ID_D), S_{ID_S})$, H(.) is one cryptographic hash function, such as MD5 [12]. Then **S** makes $M_S = [ID_S, ID_D, g^a, H(g^a||K_{SD}||0)]$ where ID_S is the real identity of **S** and ID_D is the real identity of **D**, and uses **D**'s public key to encrypt M_S as $C_S = E_{ID_D}(M_S)$ using any IBE scheme such as Hybrid-IBE in [15]. **S** also sets the number of hops from **S** to **D** as `HopCount`, which indicates the maximum hops the **ARREQ** allowed in the network. If this field contains a value of zero, the **ARREQ** must be discarded.

Afterwards, **S** broadcasts an **ARREQ** formatted as follows to all its neighbor nodes:

$$\textbf{ARREQ} = < rt\_seqno, HopCount, C_S >$$

In the end, **S** adds the entry $|rt\_seqno|ID_D|N/A|?|T_S|$ to its local route table as shown in Figure 1.

In Figure 1, the first field records the route sequence number for this route. The second field records the real identity of the destination. The third field records its upstream node of the route, which is N/A since **S** itself is the source of this route. The fourth field records its downstream node of the route, which will be added later during the path reverse phase. The last field T_S is the timer of the route, which starts when the entry is added. The timer is activated whenever no packet is launched corresponding to this route. Once the timer reaches zero, the route is simply deactivated by removing the field in **S**.

Step 2. Upon receiving the **ARREQ**, a node goes through the following procedure:

- Check if it is from one of its trusted neighbor nodes based on its sender's address, and if so, it continues. Otherwise, it stops.
- Check if the **ARREQ** has already been received from any neighbor node using rt_seqno for this route. If the **ARREQ** is fresh, it continues; otherwise, it stops.
- Check if the node is the destination by decrypting C_S with the private key of the node. If the decrypted result is meaningful, i.e. g^a is correct and the receiver's identity is the node's identity, the node is the receiver; otherwise, the node is NOT the receiver.
- If the node is NOT the intended receiver and $(\texttt{HopCount} --) \geq 0$, then it forwards **ARREQ** to all its neighbors via broadcasting, where `HopCount` in

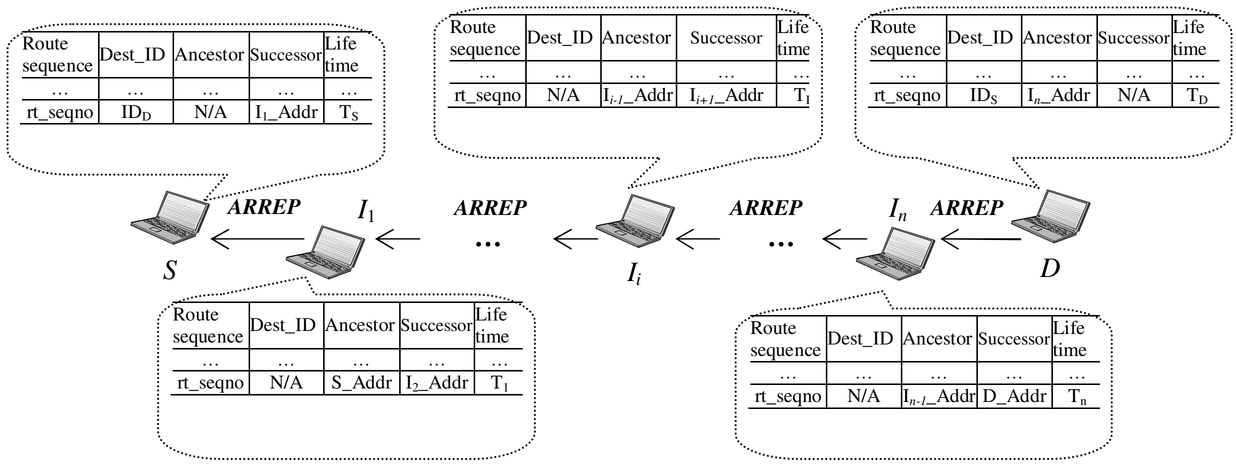

Fig. 2. Route Tables during Route Reverse Phase

ARREQ is decreased by one. In the end, the node adds the following entry to its local route table. (See Figure 1.)

$$\begin{cases} |\texttt{rt\_seqno}|N/A|\texttt{S\_Addr}|?|T_1|, \text{if the node is } I_1; \\ |\texttt{rt\_seqno}|N/A|I_{i-1}\_Addr|?|T_i|, \text{if the node is } I_i; \\ |\texttt{rt\_seqno}|N/A|I_{n-1}\_Addr|?|T_n|, \text{if the node is } I_n. \end{cases}$$

- If the node is the intended receiver **D**, it can correctly recover M_S and parse it as ID_S, ID_D, g^a and $H(g^a||K_{SD}||0)$. Then, it can verify the source **S** by checking $H(g^a||K_{SD}||0)) = H(g^a||\hat{e}(H(ID_S), S_{ID_D})||0)$, since only **S** and **D** can calculate $K_{SD} = \hat{e}(H(ID_D), S_{ID_S}) = \hat{e}(H(ID_S), S_{ID_D})$, the destination is ensured that the **ARREQ** is from the right source **S**, which means the destination is communicating with the desired node and not with some bogus nodes. **D** then adds $|\texttt{rt\_seqno}|ID_S|I_n\_Addr|N/A|T_D|$ to its local route table as shown in Figure 1. Because **D** is the destination of this route, the second last field will be set N/A, and the path discovery phase ends.

4) Route Reverse Phase: In this phase, destination **D** should respond to source **S** in the reverse path.

Step 1. **D** first randomly selects $b \in [1, p-1]$, and computes g^b and $H(g^b||K_{SD}||1)$. It makes $M_D = [ID_S, ID_D, g^b, H(g^b||K_{SD}||1)]$. D then uses **S**'s public key to encrypt M_D as $C_D = E_{ID_S}(M_D)$. In the end, it looks up its upstream node I_n according to rt_seqno, and appends an authentication tag by applying HMAC function on the encrypted message C_D together with rt_seqno using the secret key K_{DI_n} shared with the upstream node I_n. Afterwards, **D** sends an *anonymous route reply* (**ARREP**) message to I_n, which is formatted as follows:

$$\textbf{ARREP} = < rt\_seqno, C_D, MAC_{K_{DI_n}}(rt\_seqno, C_D) >$$

Furthermore, **D** also computes the shared session key $SK_{SD} = (g^a)^b$.

Step 2. When node I_n receives **ARREP** from **D**, it first uses its shared secret key K_{DI_n} to verify $MAC_{K_{DI_n}}(rt\_seqno, C_D)$ contained in **ARREP** from **D**. It drops this **ARREP** if the authentication fails. Otherwise, it continues. Further, if rt_seqno is found in its local route table, it continues; otherwise, it stops. I_n looks up its upstream node I_{n-1} from its route table. Afterward, I_n uses the shared secret key with I_{n-1} to calculate a new authentication tag on encrypted message C_D together with rt_seqno, and replace the old authentication tag with the newly created one. Finally, I_n forwards modified **ARREP** to I_{n-1}. Then, in the entry corresponding to rt_seqno, I_n updates the fourth and last fields with **D**'s address along with a new timer T_n. All the other intermediate nodes $I_1, I_2, \cdots, I_{n-1}$ along the route make the same operations as the node I_n. (See Fig. 2.) In the end, the node I_1 forwards **ARREP** to the source **S**.

Step 3. When source **S** receives **ARREP** from I_1, it first uses its shared secret key K_{DI_1} to verify $MAC_{K_{DI_1}}(rt\_seqno, C_D)$ contained in **ARREP** from I_1. It drops this **ARREP** if the authentication fails. Otherwise, it continues. Then, according to rt_seqno, **S** looks up the corresponding entry in its route table. If the entry is found, it continues; otherwise, it stops. In the found entry, **S** updates the fourth and the last fields with I_1_Addr along with a new timer $\texttt{T}_S$, respectively. On the other hand, **S** also uses its private key S_{ID_S} to recover M_D and parses it as ID_S, ID_D, g^b and $H(g^b||K_{SD}||1)$. Because only **S** and **D** can calculate the shared secret key $K_{SD} = \hat{e}(H(ID_D), S_{ID_S}) = \hat{e}(H(ID_S), S_{ID_D})$, **S** can authenticate the destination **D** by checking $H(g^b||K_{SD}||1)) = H(g^b||\hat{e}(H(ID_D), S_{ID_S})||1)$. If **D** passes the authentication, **S** then computes the shared session key $SK_{SD} = (g^b)^a$. In this way, not only the route from the source to the destination but also a shared session key SK_{SD} between them can be established.

The data packet transmission could start immediately after the route between the source and destination node is built.

5) Data Forwarding phase: In this phase, the source **S** begins to send a confidential M to the destination **D**.

Step 1. **S** uses session key SK_{SD} to encrypt M as $C = e_{SK_{SD}}(M)$. Then, it finds its downstream node I_1 from its local route table based on the identity of the destination of data packet and uses the shared secret key between them to generate a message authentication code on encrypted message C as $MAC_{K_{SI_1}}(C)$, and encrypt rt_seqno as $R_{I_1} = e_{K_{SI_1}}(rt\_seqno)$. In the end, it sends $(R_{I_1}, C, MAC_{K_{SI_1}}(C))$ to the node I_1.

Step 2. When each intermediate node I_i receives $(R_{I_i}, C, MAC_{K_{I_{i-1}I_i}}(C))$ from its upstream node, where I_{i-1} is **S** if i is equal to 1 and I_i is **D** if i is equal to $n+1$, it first uses its shared secret key $K_{I_{i-1}I_i}$ to verify $MAC_{K_{I_{i-1}I_i}}(C)$. It drops this message if the authentication fails. Otherwise, it continues and decrypts R_{I_i} for rt_seqno. Then, according to rt_seqno I_i finds the corresponding entry in its route table. Later, I_i uses the shared secret key with its successor node I_{i+1} to generate a new authentication tag on C and encrypt rt_seqno, and replace the old ones with the new ones respectively. I_i forwards $(R_{I_{i+1}}, C, MAC_{K_{I_iI_{i+1}}}(C))$ to its downstream node. At the same time, it updates its timer T_i in the last field. In this way, the node I_n will send $(R_D, C, MAC_{K_{I_nD}}(C))$ to the destination **D**.

Step 3. When the destination **D** receives $(R_D, C, MAC_{K_{I_nD}}(C))$ from I_n, it uses its shared secret key K_{I_nD} to verify $MAC_{K_{I_nD}}(C)$. It drops this message if the authentication fails. Otherwise, it continues decrypting R_D for rt_seqno. Furthermore, it finds the corresponding session key $SK_{SD} = g^{ab}$, which is taken to recover M at destination **D**. Similarly, **D** can also send confidential data to **S** in the same way.

IV. Enhancing Anonymity via Decoy Mechanism

In this section, we first introduce a new type of attack on anonymous services, called *snare attack*. Then, we present a DECOY mechanism as a countermeasure to deceive or disrupt the effort of tracing the VIN by an attacker.

A. Snare Attack

In MANETs, it may happen that a mobile node is compromised. For example, in a battlefield, a node could be compromised when the corresponding soldier is caught by the enemy. Afterward, the compromised node could be used to lure a VIN, such as the commander, into communicating with it. Since the adversary can easily intercept and eavesdrop any transmission in the network through the compromised node, the adversary can identify the physical location of the VIN by tracing and analyzing some routes. After locating the VINs, the adversary will be able to launch a *Decapitation Strike* on those VINs as a short cut to win the battle. Therefore, it is necessary to develop a countermeasure against the *snare attack*. The DECOY mechanism, which will be introduced in the next subsection, will be proposed for this purpose to enhance anonymity of the VINs.

B. DECOY mechanism

A decoy is usually a person, device or event meant, which is taken as a distraction to conceal what an individual or a group might be looking for. In reality, in case of assassins, a *very important person* (VIP) is usually protected with dozens of decoys, i.e. VIP impersonators. For a DECOY mechanism in MANETs, several mobile nodes can serve as Decoys in order to protect the VIN.

During the deployment of MANETs, a VIN (denoted as **V** in the following context) could choose n nodes as its Decoys, namely D_1, D_2, ..., D_n. Each Decoy shares a secret key $s_i, 1 \leq i \leq n$ with the VIN. Upon receiving a route request from a legitimate node **S** to **V**, **V** may randomly choose one Decoy D_i from its Decoys to answer this request, and ask D_i to establish an active route corresponding to the request. To perform this, **V** first randomly selects $b \in [1, p-1]$, and computes g^b and $H(g^b||K_{SV}||1)$. It makes $M_V = [ID_S, ID_V, g^b, H(g^b||K_{SV}||1)]$. It then computes the shared session key $SK_{SV} = g^{ab}$. Afterward, **V** uses the secret key S_i shared with the chosen Decoy D_i to encrypt (rt_seqno, M_V, SK_{SV}) and combine it with HopCount and the identities of **S** and D_i as the *Decoy route request message* (**DRREQ**) (detailed as follows) and broadcasts it:

$$\mathbf{DRREP} =< E_{S_i}(ID_{D_i}, ID_S, rt\_seqno, M_V, SK_{SV}), HopCount >$$

Any Decoy node will check if the node is the intended receiver by trying to decrypt **DRREQ** with the secret key shared with his VIN. If the decrypted result is meaningful, i.e. the receiver's identity is the node's identity, the Decoy node is the receiver; otherwise, the node is NOT the receiver. If the node is NOT the intended receiver or non-Decoy node, it forwards **DRREQ** to its neighbors via broadcasting and decreases `HopCount` in **DRREQ** by one.

After Decoy node D_i receives **DRREQ**, it uses source **S**'s public key to encrypt M_V as $C_V = E_{ID_S}(M_V)$. In the end, it looks up its upstream node I_n according to rt_seqno, and appends an authentication tag on encrypted message C_V together with rt_seqno using the secret key $K_{D_iI_n}$ that is shared with the intermediate node I_n. Afterwards, D_i sends an *anonymous route reply message* (**ARREP**) to I_n, which is expressed as follows:

$$\mathbf{ARREP} =< rt\_seqno, C_V, MAC_{K_{D_iI_n}}(rt\_seqno, C_V) >$$

If Decoy node D_i cannot find the corresponding route entry from source **S**, it stores **DRREQ** temporally until either a right **DRREQ** is received or it expires. With the above mechanism, source **S** actually communicates with node D_i instead of VIN such that the VIN could survive through a *snare attack*.

V. Anonymous and Security Analysis

In this section, we analyze the proposed ASRPAKE protocol, especially in the aspects of (*i*) end-to-end anonymity of a route (i.e., the anonymity along all the intermediate node of the route from the source to the destination), and (*ii*) the security of the authenticated session key shared by the source and the destination.

First, ASRPAKE maintains the end-to-end anonymity of a route provided that not all the intermediate nodes along the route are in collusion. The downstream node of source **S** only knows which data originally came from the source and which data was forwarded by the source. However, the node has no idea which node is the actual receiver. On the other hand, the upstream node of destination **D** knows the actual receiver corresponding to route, but it cannot gain the identity of the source. In addition, some intermediate nodes along the route may be in collusion together. In this case, the nodes in collusion can know which data was just forwarded. However, they cannot recognize the source and destination. Therefore, unless all the intermediate nodes are in collusion, ASRPAKE can maintain strict end-to-end anonymity.

Secondly, based on the application scenarios of interest, we examine the security of ASRPAKE in terms of the following four security attributes [9]:

(1) *Known session key security*: Each run of key exchange between two entities should produce a unique secret key. Known session key security can be achieved if a run of key exchange is secure in presence of an adversary which has learned some previous session keys. In view of the randomness of a and b in ASRPAKE, session keys in different runs of key exchange are independent of each other. The knowledge of previous session keys does not help an adversary to derive any future session key $SK_{SD} = g^{ab}$.

(2) *Forward secrecy*: It can be achieved if secrecy of previous session keys established by honest entities cannot be affected even when the long-term private keys of one or more entities are compromised. In ASRPAKE, the adversary has to solve the corresponding ephemeral keys a and b to learn the previous session key $SK_{SD} = g^{ab}$, which is a discrete logarithm problem even though the adversary has got the private keys S_{ID_S} and S_{ID_D}. Therefore, ASRPAKE can keep forward secrecy.

(3) *No key compromise impersonation*: With ASRPAKE, compromising one entity's private key does not help to compromise the private key of any other entity. Clearly, the adversary may impersonate the compromised entity in the subsequent protocol operations; however, the adversary still cannot impersonate the other entities because it has no idea of the private keys of them.

(4) *No unknown key share*: The network suffers from an unknown key share attack in the case that an adversary has successfully convinced an entity that the entity shares a specific session key with another entity, while in reality the entity shares the key with the adversary. ASRPAKE can defeat such an attack since an adversary needs to learn the private key of the source node before the static secret key shared with the destination node can be solved. Note that without the static secret key of the destination, the attack can hardly work.

In summary, the proposed protocol, ASRPAKE, can maintain strict end-to-end anonymity, and the session key established in our proposed ASRPAKE protocol is secure.

VI. Conclusions

In this paper, we have presented a novel anonymous secure routing protocol, ASRPAKE, with a suite of embedded authenticated key exchange mechanisms for MANETs. The main features include (*i*) the achievable end-to-end anonymity and security; (*ii*) the integration of the authenticated key exchange operations into the routing algorithm. Furthermore, a new type of attack on anonymous services called *snare attack* was introduced, by which an adversary could snare the VIN and launch a *decapitation strike* on a VIN. As a countermeasure to the *snare attack*, we also presented a novel DECOY mechanism to enhance anonymity of VINs and defeat the *snare attack*. As the future research, we plan to improve route efficiency while preserving the security and anonymity, such as secure position aided routing to avoid route messages flooding.

References

[1] B. Zhu, Z. Wan, M. Kankanhalli, F. Bao and R. H. Deng, "Anonymous secure routing in mobile ad-hoc networks", in *Proc. of LCN'04*, pp. 102-108, 2004.

[2] R. Song, L. Korba, G. Yee, "AnonDSR: efficient anonymous dynamic source routing for mobile ad-hoc networks", in *Proceedings of SASN*, pp. 32-42, Alexandria, Virginia, USA. November 7-11, 2005.

[3] A. Boukerche, K. El-Khatib, L. Xu and L. Korba, "An efficient secure distributed anonymous routing protocol for mobile and wireless ad hoc networks", *computer communications*, Vol. 28, pp. 1193-1203, 2005.

[4] Y. Zhang, W. Liu, W. Lou, Y. Fang, "MASK: anonymous on-demand routing in mobile ad hoc networks," *IEEE Transactions on Wireless Communications*, Vol. 5, No. 9, pp. 2376-2385, 2006.

[5] V. S. Miller, "Use of Elliptic Curves in Cryptography," *Advances in Cryptology*, pp. 417-426, August 18-22, 1985

[6] J. Herranz, G. Saez, "Forking lemmas for ring signature schemes", *Proceedings of Indocrypt 2003. T. Johansson and S. Maitra (Eds.), Lecture Notes in Computer Science 2904*, pp. 266-279, 2003. Also see: http://eprint.iacr.org/2003/067.pdf

[7] Y. Zhang, W. Liu, W. Lou, Y. Fang, "Location-based compromise-tolerant security mechanisms for wireless sensor networks," *IEEE Journal on Selected Areas in Communications, Special Issue on Security in Wireless Ad Hoc Networks*, Vol. 24, No. 2, pp. 247-260, 2006.

[8] J. Broch, D. A. Maltz, D. B. Johnson, Y. C. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols", in *Proc. ACM Mobicom 1998*, 1998.

[9] S. Blake-Wilson and A. Menezes,, "Authenticated Diffie-Hellman key agreement protocols," *Proceedings of the 5th Annual Workshop on Selected Areas in Cryptography (SAC' 98)*, LNCS 1556, Springer-Verlag, pp. 339-361, 1999.

[10] C. Boyd, W. Mao and K. Paterson, "Key agreement using statically keyed authenticators", in *Applied Cryptography and Network (ACNS 2004)*, LNCS 3089, Springer-Verlag, 2004.

[11] W. Mao, *Modern Cryptography: Theory and Practice*, Prentice Hall PTR, 2003.

[12] B. Schneier, *Applied Cryptography* (2nd), John Wiley: New York, 1996.

[13] C. E. Perkins, E. M. Royer, "Ad-hoc on-demand distance vector routing," *In Proc. of Second IEEE Workshop on Mobile Computer Systems and Applications*, 1999.

[14] D. B. Johnson, D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," *Mobile Computing*, 1996.

[15] B. Libert, J. J. Quisquater, "Identity based encryption without redundancy," *Applied Cryptography and Network Security: Third International Conference, ACNS 2005*, New York, NY, USA, June 7-10, 2005.

[16] The Network Simulator - ns-2. *Available at http://nsnam.isi.edu/nsnam/index.php/User_Information*

[17] Multiprecision Integer and Rational Arithmetic C/C++ Library (MIRACL), *Available at http://indigo.ie/ mscott/*

[18] S. Tzu, "Thirty-six strategies," *Available at http://en.wikipedia.org/wiki/Thirty-Six_Strategies*