This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the WCNC 2007 proceedings.

Real-Time Anonymous Routing for Mobile Ad Hoc Networks[†]

Jung-Chun Kao and Radu Marculescu
Carnegie Mellon University
Pittsburgh, PA 15213-3890, USA

***Abstract*—We propose the *A*nonymous *S*ymmetrically *C*ryptographic (*ASC*) routing protocol which is entirely based on a *symmetric* cryptosystem. The ASC protocol preserves the identity privacy, location privacy and route anonymity with only a negligible overhead in terms of processing requirements and packet size. Furthermore, the ASC protocol does not rely on any trusted agent or centralized mechanism (both of which are impractical in hostile environments). Compared with other anonymous routing protocols, the ASC protocol reduces the end-to-end delay by orders of magnitude, while performing comparably well in terms of packet delivery ratio. These features enable anonymity for applications with strict QoS requirements (e.g. video and audio streaming) over mobile ad hoc networks.**

I. Introduction

Protecting personal privacy is of primary concern for emerging mobile ad hoc networks (MANET) and wireless sensor networks. Important privacy information includes identity, location and communication content. As an important component of privacy, *user anonymity* can improve security significantly by making adversaries unable to identify potential victims and conduct target-specific attacks.

However, due to several factors like wireless medium, node mobility, lack of infrastructure, etc., preserving anonymity is a complex issue in ad hoc wireless networks. An adversary (or a set of adversaries) can easily eavesdrop packets over wireless communication channels, and then disclose the communication content and/or interfere with the normal communication in the network. In addition, due to the lack of infrastructure, the source and destination nodes need to rely on the intermediate nodes to relay their messages. This makes the nodes even more susceptible to attacks and motivates the study of anonymous routing in MANET and wireless sensor networks.

Cryptographic techniques are the most effective way for anonymous routing algorithms to establish a line of defense against malicious attacks. In general, an anonymous routing algorithm relies on *link encryption*—which makes data forwarding untraceable by shuffling the packets at each link—and *path encryption*—which encrypts the communication content so that only the intended recipient can decrypt the encrypted message. However, performing cryptographic operations incurs a considerable processing overhead. In many cases, the execution times of encryption and decryption dominate the performance of anonymous routing protocols and the overall system cost.

Related Work

There exists related work on anonymous routing algorithms in ad hoc wireless networks. Kong and El-Khatib respectively proposed ANODR (*AN*onymous *O*n *D*emand *R*outing) [1] and SDDR (*S*ecure *D*ynamic *D*istributed *R*outing) [2]. However, Zhu [3] categorized anonymity into *identity privacy*, *location privacy*, and *route anonymity*, and pointed out the failures of the above approaches in preserving the anonymity. The *A*nonymous *S*ecure *R*outing (ASR) protocol proposed by Zhu [3] satisfies all the three requirements of anonymity. However, ASR is inefficient while establishing the route due to a considerable number of asymmetric cryptographic operations.

More recently, Wu and Bhargava [4] proposed AO2P (*A*d Hoc *O*n-Demand *P*osition-Based *P*rivate Routing) whose performance benefits greatly from the information of nodes' geographical position. However, according to Papadimitratos and Haas [11], the intrinsic requirements of AO2P—node authentication and position servers—are *not* available in MANET.

All of the above protocols are based on *asymmetric cryptosystems*. However, the asymmetric cryptosystems have a few intrinsic disadvantages: large computational latency, key size, and power consumption. Table 2 in [1] clearly shows the huge processing overhead incurred by the asymmetrically cryptographic operations. Besides this, the asymmetric cryptography entails a major performance overhead due to the increase in the packet size (see Table 2 in [7]). Indeed, the experience learned with a low-end system, namely TinySec [5], reveals that even using dedicated hardware reduces only the computational costs, but not the packet size. Furthermore, the asymmetric cryptographic operations are typically power hungry [15] and beyond the capabilities of today's sensor networks [6].

Very importantly, *none* of the above anonymous protocols are fast enough to route real-time traffic; this is because they rely on slow and expensive *asymmetric cryptography*. A software-based RSA cipher takes hundreds of milliseconds *per link* to encrypt and decrypt a block (see Table 2 in [1]). Nevertheless, the maximum one-way end-to-end delay acceptable for real-time traffic is typically 150ms [8]. Because the end-to-end delay is greater than the product of the execution time of link encryption per link and the number of links in the path, even a dedicated hardware-based RSA cipher [9], with a 30× speed-up with respect to the software-based cipher in [1], cannot meet the imposed deadline. All these issues motivate our research and justify the advantages of the approach we propose in this paper.

Our contribution

In contrast to previous work, our proposed ASC protocol is based on a *symmetric* cryptosystem; any other anonymous routing protocol available to date requires an asymmetric cryptosystem for path encryption, link encryption, or both, to satisfy anonymity the requirements. Compared to an asymmetric cryptosystem, a symmetric cryptosystem typically brings a 4-order-of-magnitude speed-up [10] and a significant decrease in key length. Such a high speed and low overhead makes the ASC protocol fast enough to route real-time traffic in a timely fashion. This distinguishes our protocol from other anonymous routing protocols in the literature.

Another distinctive feature is that, to the best of our knowledge, the ASC protocol is the *first* anonymous routing protocol which uses an adaptive power scheme for transmission. Our simulation results show that, in terms of the packet delivery ratio, a simple

[†] This research is supported by CMU CyLab ARO under grant no. 9097.60.5 and by a Frank J. Marshall Graduate Fellowship for J.-C. Kao.

transmission power scheme with three levels of transmission ranges outperforms the constant transmission power scheme.

In addition to the huge speedup due to symmetrically cryptographic operations and the improvement of packet delivery ratio due to the adaptive transmission power scheme, the ASC protocol provides the following three lines of defense:

- First, ASC uses an adaptive transmission power scheme for improving the network security. This scheme first attempts to transmit the packets at the lowest transmission power. If the packets cannot reach their destinations by a predefined deadline, the ASC protocol *gradually* increases the level of transmission power, thus reducing the eavesdropping risk [16]. Since adversaries can hear only a small portion of packets, the probability of privacy exposure is significantly decreased.
- Second, the path encryption (with the help of a *symmetric* cryptosystem) makes packets unreadable to all nodes except the intended recipients. This is because a packet is encrypted with a session key *before* being transmitted and only the destination node can decrypt the packet using the session key.
- Third, in order to make a node/route untraceable, each intermediate node shuffles the payload and the header of a packet by using link encryption and virtual circuit identifiers, respectively. This eliminates all the *common information* appearing in the packet[1] while not increasing the time to take the routing decision.

Combining the above three defense lines, the ASC protocol is able to preserve anonymity and defend against various passive and active attacks. Note that both path encryption and link encryption use the same symmetric cryptosystem but with different keys. As justified in [11] and [12]-[14], respectively, the keys used for path encryption and link encryption can be established *without* running asymmetric cryptographic operations at run time, for instance, by using a secure key exchange [18], an initial distribution of credentials, or a self-configured key management scheme [14].

Although the focus of this paper is *not* on time analysis attacks [17], we note that it is possible (even on resource-constrained devices) to add to the ASC protocol countermeasures like dummy traffic injection and padding. This is because compared to asymmetric cryptosystems, the huge savings in computation and communication resources due to symmetric cryptosystems can be used for further security improvements.

The remainder of this paper is organized as follows. Section 2 introduces the basic notations and underlying assumptions. Section 3 describes our proposed ASC protocol. The simulation results are shown in Section 4. Finally, Section 5 presents concluding remarks and future work.

II. Notations and Underlying Assumptions

We define $e_k(\cdot)$ as the ciphertext encrypted with the key k and $d_k(\cdot)$ as the plaintext decrypted with the key k. Since the ASC protocol uses a symmetric cryptosystem, $d_k(e_k(x)) = x$ holds for every possible plaintext x and every possible key k.

As shown in Figure 1, we denote the source node, the intermediate nodes and the destination node by S, X_i ($i = 1, 2, \ldots, h$-1) and D, respectively, where h is the number of hops between the source and the destination. The source and the destination are also denoted by X_0 and X_h, respectively. For clarity, Figure 1 does not include the neighboring nodes which may create routing loops and redundant packets, but as discussed in Section III, the ASC protocol can deal with these two problems too.

[1] Except the "type of service" field which does not reveal any useful information.

Each node has a unique identity which must be kept secret from adversaries and intermediate nodes since otherwise, identity privacy is compromised. As discussed later, the *security association* (*SA*) between the source S and destination D is represented by a shared key D^*. This key is only used for discovering a route. After a route is established, the source and destination agree to use a session key, denoted by K_{SD}, for path encryption.

As to link encryption, we define $L_{i,j}(\beta)$ as the link key shared by the neighboring nodes X_i and X_j, where β is any key seed. When receiving a shuffled packet, an intermediate node first recovers the contents of the packet. After processing it, the intermediate node may either drop or shuffle the packet with a link key before forwarding it. Fig. 1 illustrates the hop-by-hop shuffling process.

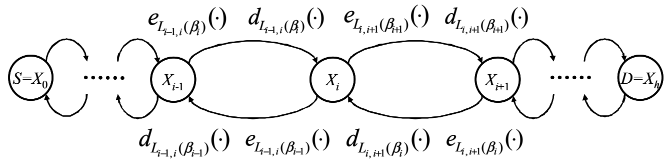
$e_{L_{i-1,i}(\beta_i)}(\cdot)$
$d_{L_{i-1,i}(\beta_i)}(\cdot)$
$e_{L_{i,i+1}(\beta_{i+1})}(\cdot)$
$d_{L_{i,i+1}(\beta_{i+1})}(\cdot)$
$S=X_0$
X_{i-1}
X_i
X_{i+1}
$D=X_h$
$d_{L_{i-1,i}(\beta_{i-1})}(\cdot)$
$e_{L_{i-1,i}(\beta_{i-1})}(\cdot)$
$d_{L_{i,i+1}(\beta_i)}(\cdot)$
$e_{L_{i,i+1}(\beta_i)}(\cdot)$

Fig. 1. Before forwarding a packet, each intermediate node shuffles the packet using encryption with a link key. The receiver recovers the shuffled packet and then it shuffles the packet again with a different link key before forwarding it. Shuffling with distinct link keys makes packets untraceable.

We assume the existence of two classes of security associations (*SA*s): the *end-to-end SA* (*i.e.* between the source nodes and the destination nodes) and *link-based SA* (*i.e.* between neighboring nodes). The end-to-end SA helps to establish a route between source and destination and determine the *session key* used in path encryption. The link-based SA helps to establish *secure links* and decide the actual *link key* used in link encryption.

As justified in [11] and [12]-[14], the end-to-end and link-based SAs can be established *without* running asymmetric cryptographic operations at run time. For example, they can be achieved with a secure key exchange [18], an initial distribution of credentials [12]-[13], or a self-configured key management scheme [14]. Depending on the desired level of security and/or resource constraints, the end-to-end SA can be pairwise, per node basis, or (in general) per group basis. Although the end-to-end SA brings a computational overhead linear to the number of groups, the 4-order-of-magnitude acceleration gained by using symmetric cryptosystems can easily overcome this overhead. This makes the end-to-end SA widely used in secure protocols (like [11]) and anonymous protocols (like ASR [3]).

The end-to-end SA between the source S and destination D is represented by a shared key D^*. After a route is established with the help of this key D^*, the session key used for path encryption is generated. This session key is denoted by K_{SD}. As we will discuss in Section III, a *challenge-response* mechanism is used in the ASC protocol to generate the session key.

The existence of the link-based SA is valid because it is the very basis of a key management problem. Although any of key management schemes can be used, practical limitations make random key predistribution schemes (e.g. [12]-[13]) and self-configured schemes [14] attractive because they do not rely on asymmetric cryptographic operations or trusted agents.

For simplicity of exposition, the link-based SA between node X_i and node X_j is represented by a shared key $L_{i,j}$. The actual link key used in link encryption is denoted by $L_{i,j}(\beta)$ where β is a *key seed*. Two simple examples of $L_{i,j}(\beta)$ are $L_{i,j}$—using $L_{i,j}$ itself as the link key—and $e_{L_{i,j}}(\beta)$—using the encrypted form of the key seed as the link key. The latter one provides better flexibility and security. Even when a link-based SA is compromised, an adversary still cannot break the link encryption unless the dynamically assigned key seed is also intercepted. Because the adaptive transmission power scheme in the ASC protocol tends to reduce the probability of packets being intercepted, our ASC routing protocol uses the latter scheme to further improve security and anonymity.

III. The ASC Routing Protocol

The ASC routing protocol is connection-oriented. To transmit data, the protocol consists of the following phases: route request, route acceptance, data transmission, and route maintenance. In this section, we present the detail of each phase and also the underlying medium access control (*MAC*) mechanism.

A. MAC Mechanism

Once a source needs to find a route to its destination, it first broadcasts a route request (*RR*) frame. Because of not knowing a priori *who* and *where* to forward the packet, the receiver address in the MAC header is set to 0 to denote broadcasting. (As discussed later in Section III.B, the RR frames do not flood the entire network because *i)* the ASC protocol reduces the dissemination region by using the lowest possible transmission power and *ii)* the time-to-live mechanism restricts the lifetime of frames.) All the nodes within the transmission range remove the MAC header and the tail of the frame and then process it at the network layer in the protocol stack. The Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) channel access mechanism is used for a RR frame; that is, before a RR frame is sent, the channel has to be sensed idle for a distributed inter-frame space (DIFS) of time.

During the route discovery phase, the RR frame is duplicated and broadcasted via several intermediate nodes until one of them reaches the destination node. Due to the *adaptive transmission power scheme*, we assume multiple levels of transmission ranges. Figure 2 shows the generic frame format since all types of frames only differ in the field of frame body. The frame body fields will be detailed in Section III.B-III.D. One should note that the sender and receiver addresses in the MAC header can be either real MAC addresses or *pseudo* MAC addresses to further protect anonymity.

Once the destination receives the RR frame, it generates a route acceptance (*RA*) frame. The RA frame will be forwarded reversely along the same route in which the corresponding RR frame was transmitted. In each hop, RA frames are transmitted using the distributed coordination function (DCF) of IEEE 802.11 (i.e. CSMA/CA plus RTS/CTS) as the MAC mechanism to alleviate packet collision. Similarly, data (*DA*) frames are transmitted along the same route by using the same DCF of IEEE 802.11.

We note that IEEE 802.11 DCF can reduce the occurrence frequency of packet collisions caused by the hidden terminal problem, but *cannot* completely eliminate it. Such a packet collision will eventually trigger the timeout mechanism in the ASC protocol and the corrupted packet will be resent. Unless the channel contention is so severe that the connection deadline is exceeded, this guarantees a successful transmission.

	← MAC header			→		
Bytes	2	2	6	6		4
	Frame control	Duration	Sender addr.	Receiver addr.	Frame body	CRC

Fig. 2. Generic format of frames in the ASC routing protocol. Three major types of frames—RR, RA and DA frames—only differ in the field of *frame body* as detailed in Section III.B, III.C and III.D.

In the following subsections, we focus on the routing mechanism at the network layer.

B. Route Request Phase

To establish a route to the destination node D, the source node S first broadcasts a route request (*RR*) packet and sets the round-trip timer (t_{RTT}) and the connection timer ($t_{connect}$). The RR packet is transmitted using the *adaptive transmission power scheme*, which first sends a packet at the lowest level of transmission power and, if an acknowledgement message does not return in a predefined time interval, it resends it using higher transmission power.

Initially, the time-to-live (*TTL*[2]) field of the RR packet is set to a small value and the suggested level of transmission power (P) is set to the lowest level. All intermediate nodes relay the RR packet at the suggested power level. The intermediate nodes do not decide their transmission power on their own because this way allows the source node have a better control on the one-way latency in order to meet the firm deadlines of real-time traffic.

Sending packets at the lowest power has several advantages: saving transmission energy, reducing the eavesdropping risk [16], reducing packet collisions, and not flooding this request over the entire network. However, it is possible that the destination cannot be reached. A time-out mechanism is used to solve this problem. When the t_{RTT} timer expires, the source resets the t_{RTT} timer and generates a new RR packet with a different session number, a larger TTL value and a higher transmission power.

If the destination node is within coverage, the source node will enter the data transmission phase after receiving a valid route acceptance packet by the expiration of timer $t_{connect}$. Otherwise, the timeout of timer $t_{connect}$ will terminate all unconnected requests ending at node D. In such as case, an exception is reported to the higher-level layer which will determine the next action.

During the route request phase, an intermediate node, denoted by X_i, where $i = 1, 2, \ldots, h$, receives a route request packet with the following format:

$$[\ RR, I_{i-1}, e_{D^*}(D, ssn), P, TTL, \beta_{i-1}\ ]$$

where

I_{i-1}	The *virtual circuit identifier* (*VCI*) assigned by sender X_{i-1}
$e_{D^*}(\cdot)$	The encryption function with encryption key D^*, which is the secret key shared by the source S and destination D
D	The destination node
ssn	The session number
TTL	The time-to-live
P	The suggested level of transmission power

[2] TTL is the maximum number of hops before a packet should be discarded.

β_{i-1} The seed of the link key used to restore the shuffled payloads of incoming packets at node X_{i-1}

After receiving the RR packet, node X_i first checks whether or not there exists an entry in its routing table which has the same field $e_{D^*}(D, ssn)$ but a larger or equal level of transmission power P. This is needed to avoid looping and processing multiple instances of a single route request. If there is such an entry in the routing table, node X_i simply discards the packet and skips any further processing. Otherwise, node X_i attempts to extract its unique identity by decrypting $e_{D^*}(D, ssn)$ with its secret key X_i^*.

If successful, node X_i realizes that it is actually the destination node. After recording the entry $(I_{i-1}, 0, e_{D^*}(D, ssn), P, \beta_{i-1}, \beta_i, 0)$[3] into its routing table, it enters the route acceptance phase. Otherwise, node X_i creates a new VCI I_i, decreases the TTL by one, selects the seed β_i for future hop-by-hop shuffling, stores the entry $(I_{i-1}, I_i, e_{D^*}(D, ssn), P, \beta_{i-1}, \beta_i, \sim)$[4] in its routing table, and broadcasts the following RR packet using the same suggested level of transmission power P as follows:

$$[\ RR, I_i, e_{D^*}(D, ssn), P, TTL, \beta_i\ ]$$

The broadcasted RR packet will be received and processed the same way by the neighboring nodes until it finally arrives at destination node D, of course, if the destination is reachable under the current TTL and level of transmission power.

Note that although there are several fields remaining unchanged during the RR phase, such common information does *not* cause anonymity concerns because the route request is a broadcasting process by its nature and a limited number of adversarial nodes cannot trace these dispersing packets towards the destination node. Unlike the RR phase, common information during the route acceptance phase and data transmission phase does cause anonymity concerns and must be prevented. The details will be presented in the following sections.

C. *Route Acceptance Phase*

Once the destination node D receives the RR packet, it extracts the session number ssn using its secret key (K_{D^*}), chooses a random number as the session key K_{SD} (which will be used for path encryption during this new session), and computes $e_{K_{SD}}(ssn)$ as the acknowledgement for the route establishment (and also as the response for the challenge-response mechanism initiated by the source). Node D also selects a random number as the key seed β_h for the link key. After encrypting with the key K_{D^*} and shuffling with the link key $L_{h-1,h}(\beta_{h-1})$, node D sends out the route acceptance (RA) packet.

During route acceptance, the intermediate node X_i receives a route acceptance (RA) packet with the following format:

$$[\ RA, I_i, \beta_{i+1},\ e_{L_{i,i+1}(\beta_i)}(e_{D^*}(K_{SD}, e_{K_{SD}}(ssn)))\ ]$$

Node X_i first checks whether or not there exists an entry with the matched VCI in the routing table (*i.e.* looking for an entry whose second field is equal to I_i). If such entry does not exist, node X_i simply discards the packet and skips any further processing. Otherwise, node X_i updates the last field of that entry with the third field of the received RA packet, β_{i+1}, so that the entry in the routing table becomes $(I_{i-1}, I_i, e_{D^*}(D, ssn), P, \beta_{i-1}, \beta_i, \beta_{i+1})$. Meanwhile node X_i recovers the shuffled payload (*i.e.* the fourth field in the packet format indicated above) of the RA packet with the key $L_{i,i+1}(\beta_i)$ and then shuffles the payload with a different key $L_{i-1,i}(\beta_{i-1})$. After replacing the VCI field of the RA packet with the first field of that entry (I_{i-1}), node X_i relays to node X_{i-1} the following RA packet:

$$[\ RA, I_{i-1}, \beta_i,\ e_{L_{i-1,i}(\beta_{i-1})}(e_{D^*}(K_{SD}, e_{K_{SD}}(ssn)))\ ]$$

The relayed RA packet is received and processed the same way by the nodes en route. Finally, it arrives at the source node S which extracts the session key K_{SD} and checks the correctness of the received response, $e_{D^*}(e_{K_{SD}}(ssn))$. If correct, then the source node S starts the data transmission phase. Otherwise, it discards the received RA packet.

We note that in the RA phase, the packet is untraceable because there is no common information (except for the type of service). The fields are either replaced by different numbers (e.g. different VCIs and different seeds of the link keys), or shuffled by link encryption (e.g. the remaining fields). In addition, real identities and location information are not used. The above arguments guarantee identity privacy, location privacy and route anonymity.

D. *Data Transmission Phase*

Once the source node receives a successful RA packet, the source starts sending the data packets. A data packet is shuffled at each intermediate node and forwarded along the established route. The intermediate node X_i receives the packet with the following format:

$$[\ DA, I_{i-1},\ e_{L_{i-1,i}(\beta_i)}(e_{K_{SD}}(ssn, seq, ack, \text{data}, checksum))\ ]$$

If node X_i is the destination, then the forwarding process terminates. Otherwise, node X_i replaces the VCI with the corresponding one stored in the routing table and forwards the following shuffled packet:

$$[\ DA, I_i,\ e_{L_{i,i+1}(\beta_{i+1})}(e_{K_{SD}}(ssn, seq, ack, \text{data}, checksum))\ ]$$

Similar to the RA phase, identity privacy, location privacy and route anonymity are all satisfied. To deal with route failures due to nodes mobility or nodes failures, the source sets the connection-broken timer (t_{broken}) at the moment of entering the data transmission phase. Whenever the source node receives either an acknowledgement or a data packet originating from the destination node, it resets the t_{broken} timer. When the t_{broken} timer expires or a route-broken message is received, the source node goes to the route maintenance phase.

E. *Route Maintenance*

During the data transmission phase, one or more hops en route may be broken due to packet collision, mobility, or node failures. In such a case, the sender (either the source or intermediate node) will retransmit the packet via the same link because it cannot receive an acknowledgement frame by the expiration of the retry timeout. This is how the DCF of IEEE 802.11 works and no extra mechanism is needed. If the retransmission count exceeds a predefined threshold, the sender will send out a route-broken message towards the source node. If this message reaches the source node, the source node will start a new route request phase. Meanwhile, all the intermediate nodes receiving this message will discard the corresponding entries stored in their routing tables. If for any reason, the route-broken message cannot reach the source

[3] The symbols "0" and "~" mean "don't care" and "not set yet", respectively.

[4] The entry at the source node is $(0, I_0, e_{D^*}(D, ssn), P, 0, \beta_0, \sim)$.

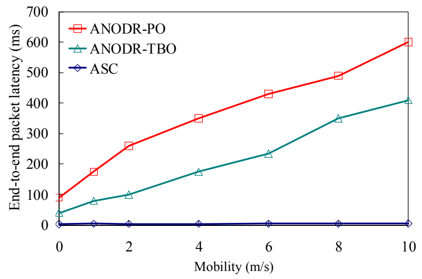

Fig. 3. End-to-end packet latency. The simulation values of ANODR-PO and ANODR-TBO are reproduced from [1]. As shown, the ASC performance exceeds the ANODR performance by orders of magnitude.

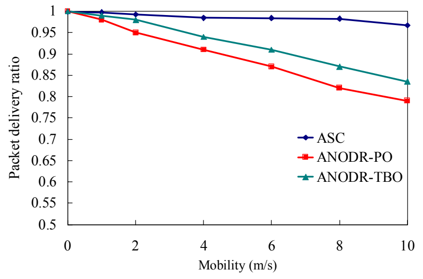

Fig. 4. Delivery ratio comparison. The simulation values of ANODR are from [1]. Unlike ANODR-PO and ANODR-TBO, as the speed of mobile nodes increases, the packet delivery ratio under ASC decreases slowly. This advantage can be attributed to its small end-to-end latency.

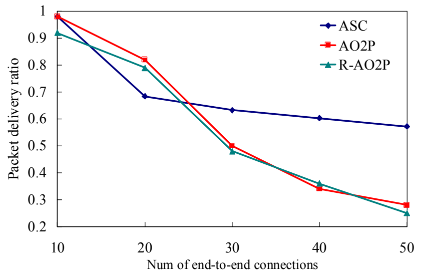

Fig. 5. Packet delivery ratio. The simulation values of AO2P are from [4]. This figure shows that when as the traffic injection rate increases, the performance gain (in terms of the packet delivery ratio) of ASC over AO2P (and its variation R-AO2P) increases.

node by the expiration of the t_{broken} timer, the timeout will trigger a new route request phase.

IV. Performance Evaluation—ASC vs. Previous Algorithms

In this section, we present two sets of comparisons, namely, *ASC vs. ANODR* and *ASC vs. AO2P*. For a fair comparison, the multiple transmission power levels in the *ASC* protocol are set to typical transmission ranges values instead of optimally customizing them. We note, however, that a full customization can clearly bring additional performance improvements for the ASC approach. The transmission ranges for the *ASC*'s adaptive transmission power scheme are set to 30.48m, 91.44m and 250m. The first two values are the typical 802.11b/g transmission ranges in office environments and outdoor, while both ANODR and AO2P use the last value as transmission range.

The values of other critical parameters are set as follows. The time durations for SIFS and DIFS are 28µs and 128µs, as in 802.11 WLAN. The back-off duration ranges from 1 to 32 slots, where a slot has a length of 50 µs. The timer t_{RTT} expires after 50ms and $t_{connect}$ expires after 150ms. Here $t_{connect}$ is set to match the maximum one-way end-to-end delay acceptable for real-time traffic. The ASC protocol uses 128-bit AES as the default encryption algorithm. The execution time of encrypting and decrypting an AES block is taken from [1].

A. Scenario I: ASC vs. ANODR

In these simulations, we follow the simulation setup described in [1]. The network field is set to 1500m × 300m. The data rate in a wireless channel is 2 Mb/s. Five short-lived source-destination pairs are maintained and continuously renewed. Each source-destination pair injects data packets of size 512 bytes at a rate of four packets per second. The node mobility speed varies between 0 to 10 m/s and the pause time is fixed to 30 seconds. Results are averaged over multiple runs with different seeds for the random number generator.

The comparisons between ASC and ANODR are shown in figures 3 and 4. Figure 3 shows that in terms of the average end-to-end delay of data packets, ASC exceeds ANODR by orders of magnitude, across all simulations. This huge gain has a simple explanation: While the bottleneck of other anonymous routing protocols available to date is due to the complexity of cryptographic operations (for example, as in [1], encrypting plus decrypting a single block once takes at least 180ms), ASC relies on a symmetric cryptosystem which is typically a few orders of magnitude faster than an asymmetric one. Such a small end-to-end latency enables the ASC protocol to route real-time traffic efficiently.

We note that under ASC, the *overall* end-to-end delay (less than 6ms under this low-traffic-rate setup) counts the waiting time and retransmission(s) due to the timeout mechanism. The timer t_{RTT} is set to expire after 50ms, which is larger than the simulated delay by orders of magnitude. In fact, the *sheer* end-to-end delay (excluding the waiting time spent on timeout) is roughly ten times smaller than the overall end-to-end delay. The above argument implies that unlike the other existing anonymous routing protocols, the slowdown of ASC is due to the routing parameters rather than cryptographic operations.

Figure 4 shows the comparison between ASC and ANODR in terms of the *average packet delivery ratio* where the simulation values of ANODR are reproduced from [1]. Under the ASC protocol, packet loss due to various reasons such as mobility, packet collision and the hidden terminal problem is considered in our simulations. The hidden terminal problem is unavoidable in this simulation setup (*i.e.* a 1500m × 300m wide network and transmission range less or equal to 250m). The DCF of IEEE 802.11 can only help to reduce the occurrence of packet collisions but *cannot* eliminate packet collisions caused by the hidden terminal problem.

The simulation results in Figure 4 also show that as the speed of mobile nodes increases, the packet delivery ratio of ASC decreases *much slower* than the packet delivery ratio of ANODR decreases. This observation implies that ASC is more suitable than ANODR in *mobile* environments like MANET. The flatter packet delivery ratio curve of the ASC algorithm is due to its significantly smaller end-to-end delay (less than 6ms). Under ASC, while a packet is on its way towards the destination, the mobile destination node at a speed of 10m/s will not move more than 10m/s * 6ms = 0.06m away from its original location. On the other hand, because the end-to-end delay under ANODR is several orders of magnitude larger, the mobile destination node is likely to move much farther away before receiving a packet. This causes a faster degradation as the speed of mobile nodes increases and explains why ASC beats ANODR in MANET.

B. Scenario II: ASC vs AO2P

In this set of simulations, we follow the simulation setup described in [4]. The network covers an area of 1000m × 1000m and there are 100 low-mobility nodes uniformly distributed over the network. Each connection injects data packets of size 512 bytes at a rate of four packets per second. The number of connections is set to a number between 10 and 50 and thus the total traffic injection rate ranges from 164Kb/s to 819Kb/s. The data rate in a wireless channel is 1 Mb/s.

Figure 5 shows the performance comparison (in terms of data packet delivery ratio) between ASC and AO2P. When the total traffic injection rate is low, ASC performs comparably to AO2P. When the total traffic injection rate is high, ASC is much more likely to successfully send the packets to the destination node. This is because AO2P attempts to send a packet to the farthest node which is close to destination at a high transmission power. This intensifies the interference, thus worsening the channel contention and packet collisions. On the contrary, ASC uses the adaptive transmission power scheme which attempts to send a packet at the lowest possible transmission power. By gradually increasing the transmission power as needed, the interference and potential packet collision is likely to be minimized.

C. Overhead Comparison in Terms of Packet Size

In this section, we compare the packet sizes of several anonymous routing protocols during the RR and RA phases. For fair comparison, cryptosystems are assumed to have comparable strengths so that the actual key sizes (as in [7]) are set to 128-bit for AES, 256-bit for ECC and 3072-bit for DH/RSA. Note that the key size has a huge impact on the overhead in terms of packet size because the size of encrypted data is an integer multiple of the block size (for any block cipher like AES) and cannot be smaller than the key/modulus size (since operating over a finite field).

During the RR and RA phases, the packet sizes under ANODR (and its variations) increase linearly with the number of hops counting from the source node. Because the sizes of RR and RA packets under ASC are constant, ASC results in a smaller increase in the packet sizes than ANODR does. On the other hand, AO2P results in a zero overhead in terms of packet sizes because it does not protect nodal positions in any manner. Its enhanced version, R-AO2P, protects the nodal positions at a much larger overhead than ASC does, because R-AO2P encrypts the positions by the Diffie-Hellman (*DH*) algorithm which requires a huge key for attaining the same level of security the 128-bit AES provides.

V. Conclusion and Future Work

The main contribution of this work is a new anonymous routing protocol (called ASC from *Anonymous Symmetrically Cryptographic* protocol) meant to route real-time traffic over mobile ad-hoc networks. To the best of our knowledge, the ASC protocol is the first anonymous routing protocol which is fast enough to route real-time traffic, while preserving identity privacy, location privacy, and route anonymity. Removing the routing bottleneck due to asymmetrically cryptographic operations accelerates by orders of magnitude both the time needed to establish a route and the average end-to-end latency.

We also point out that the ASC protocol is the first anonymous routing protocol which uses transmission power control to improve the network security and performance. As shown in the simulation results, even a simple transmission power scheme with 3-level transmission ranges improves the packet delivery ratio significantly when the packet injection rate is medium or high. Our future work will focus on transmission power control from a security and network perspective.

References

[1] J. Kong and X. Hong, "ANODR: Anonymous on demand routing protocol with untraceable routes for mobile ad-hoc networks," in Proc. ACM MobiHoc, Annapolis, USA, June 2003.

[2] K. El-Khatib, L. Korba, R. Song, and G. Yee, "Secure dynamic distributed routing algorithm for ad hoc wireless networks," in Proc. ICPP Workshops, Kaohsiung, Taiwan, Oct. 2003.

[3] B. Zhu, Z. Wan, M. S. Kankanhalli, F. Bao, and R. H. Deng, "Anonymous secure routing in mobile ad-hoc networks," in Proc. IEEE Intl. Conf. on Local Computer Networks, Tampa, USA, Nov. 2004.

[4] X. Wu and B. Bhargava, "AO2P: Ad hoc on-demand position-based private routing protocol," IEEE Trans. Mobile Computing, vol. 4, no. 4, pp. 335–348, July-Aug. 2005.

[5] C. Karlof, N. Sastry, and D. Wagner, "TinySec: A link layer security architecture for wireless sensor networks," in Proc. ACM SenSys, Baltimore, USA, Nov. 2004.

[6] A. Perrig, J. Stankovic, and D. Wagner, "Security in wireless sensor networks," Communications of the ACM, vol. 47, no. 6, pp. 53–57, June 2004.

[7] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Recommendation for key management," NIST Special Publication 800-57, May 2006.

[8] ITU-T Recommendation G.114, "One-way transmission time," Mar. 1993.

[9] S. Shimizu, H. Ishikawa, A. Satoh, and T. Aihara, "On-demand design service innovations," IBM Journal of Research and Development, vol. 48, no. 5/6, pp. 751–765, 2004.

[10] E. Shi and A. Perrig, "Designing secure sensor networks," IEEE Wireless Commun. Mag., vol. 11, no. 6, pp. 38–43, Dec. 2004.

[11] P. Papadimitratos and Z. J. Haas, "Secure routing for mobile ad hoc networks," in Proc. SCS Communication Networks and Distributed Systems Modeling and Simulation Conf. (CNDS), San Antonio, USA, Jan. 2002.

[12] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in Proc. ACM Conf. on Computer and Communications Security (CCS), Washington, DC, USA, Nov. 2002.

[13] W. Du, J. Deng, Y. S. Han, and P. K. Varshney, "A pairwise key predistribution scheme for wireless sensor networks," in Proc. ACM Conference on Computer and Communications Security (CCS), Washington, DC, USA, Oct. 2003.

[14] F. Liu and X. Cheng, "A self-configured key establishment scheme for large-scale sensor networks," in Proc. IEEE Intl. Conf. On Mobile Ad-hoc and Sensor Systems (MASS), Vancouver, Canada, Oct. 2006.

[15] D. W. Carman, P. S. Kruus, and B. J. Matt, "Constraints and approaches for distributed sensor network security," NAI Labs Technical Report, Sep. 2000.

[16] J.-C. Kao and R. Marculescu, "Eavesdropping minimization via transmission power control in ad-hoc wireless networks," IEEE Intl. Workshop on Wireless Ad Hoc & Sensor Networks (IWWAN), New York, USA, June 2006.

[17] Jean-François Raymond, "Traffic analysis: Protocols, attacks, design issues, and open problems," Intl. Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability, Berkeley, USA, July 2000.

[18] N. Asokan and P. Ginzboorg, "Key agreement in ad hoc networks," Computer Communications, vol. 23, no. 17, pp 1627-1637, Nov. 2000.