Secret Common Randomness From Routing Metadata in Ad Hoc Networks

Mohammad Reza Khalili-Shoja, George Traian Amariucai, Shuangqing Wei, and Jing Deng

***Abstract*— Establishing secret common randomness between two or multiple devices in a network resides at the root of communication security. In its most frequent form of key establishment, the problem is traditionally decomposed into a randomness generation stage (randomness purity is subject to employing often costly true random number generators) and an information-exchange agreement stage, which relies either on public-key infrastructure or on symmetric encryption (key wrapping). In this paper, we propose a secret-common-randomness establishment algorithm for ad hoc networks, which works by harvesting randomness directly from the network routing metadata, thus achieving both pure randomness generation and (implicitly) secret-key agreement. Our algorithm relies on the route discovery phase of an ad hoc network employing the dynamic source routing protocol, is lightweight, and requires relatively little communication overhead. The algorithm is evaluated for various network parameters in an OPNET ad hoc network simulator. Our results show that, in just 10 min, thousands of secret random bits can be generated network-wide, between different pairs in a network of 50 users.**

***Index Terms*— Ad hoc mesh network, dynamic source routing, common randomness, secret key establishment, minimum entropy.**

I. Introduction

AUTOMATIC key establishment between two devices in a network is generally performed either by public-key-based algorithms (like Diffie and Hellman [1]), or by encrypting the newly-generated key with a special *key-wrapping key* [2]. However, in addition to the well-established, well-investigated keying information exchange, one additional aspect of key establishment is often understated: to ensure the security of the application it serves, the newly generated secret key has to be truly random. While minimum standards for software-based randomness quality are generally being enforced [3], many applications rely on often costly hardware-based *true random generators* [4]. Sources of randomness employed by true random number generators vary from wireless receivers and simple resistors to ring oscillators and SRAM memory.

In this paper, we build upon the observation that a readily-available source of randomness is usually neglected: the network dynamics. Indeed, by their very nature, communication networks are highly dynamic and largely unpredictable. Their randomness is usually evident in easily-accessible networking metadata such as traffic loads, packet delays or dropped-packet rates. However, as the main focus of our work is on mobile ad-hoc networks (MANETs), the source of randomness we shall discuss in this paper is one that is specific to infrastructure-less networks: the routing information itself. Another interesting feature of the routing information, in addition to its randomness, is that it can easily be made available to the devices that took part in the routing process, but it is usually unavailable to those devices that were not part of the route. This idea opens the door to a whole new class of applications: with the proper routing protocol, the routing information could be used for establishing *secret common randomness* between any two devices in a mobile ad-hoc network. This common randomness could then be further processed into true common randomness, and used as secret keys.

Common randomness was pioneered in [5]–[7], where it is shown that if two parties, Alice and Bob, have access to two correlated random variables (RVs) X' and Y' respectively, (in either the source or the channel models), a secret key can be established between them through public discussions and random-binning-like (e.g. hashing) operations. The key should remain secret from an adversary eavesdropper (Eve) who overhears the public discussions, and possesses side information (in the form of a third RV Z) correlated with that available at Alice and Bob. Common-randomness-based key establishment generally consists of three phases. First, Alice and Bob have to agree on two other RVs X and Y, such that $H(X|Y) < H(X|Z)$ and $H(Y|X) < H(Y|Z)$, where $H(\cdot)$ is the standard Shannon entropy. This part is sometimes called *advantage distillation*. Next, Alice and Bob (and also Eve) sample their respective random variables a large number of times, producing sequences of values. Then Alice and Bob exchange further messages (over a public channel) to agree on the same single sequence of values – this phase is the *information reconciliation*. Finally, because the agreed-upon sequence is not completely unknown to Eve (Eve can sample her variable Z synchronously with Alice and Bob), Alice and Bob run a randomness extractor on it, to produce a secret key (a shorter sequence) which, from Eve's perspective, is uniformly distributed over its space – this is the *privacy amplification phase*. The ideas of [5] and [6] have been

Manuscript received September 27, 2015; revised February 3, 2016; accepted March 21, 2016. Date of publication April 5, 2016; date of current version May 4, 2016. This work was supported by the National Science Foundation under Grant 1320351. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Y.-W. Peter Hong.

M. R. Khalili-Shoja and G. T. Amariucai are with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011 USA (e-mail: mkhalili@iastate.edu; gamari@iastate.edu).

S. Wei is with the School of Electrical Engineering and Computer Science, Louisiana State University, Baton Rouge, LA 70803 USA (e-mail: swei@lsu.edu).

J. Deng is with the Department of Computer Science of University, The University of North Carolina at Greensboro, Greensboro, NC 27412 USA (e-mail: jing.deng@uncg.edu).

Digital Object Identifier 10.1109/TIFS.2016.2550424

1556-6013 © 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

recently applied to secret key generation in wireless systems, where secure *common randomness* is attained by exploiting reciprocal properties of wireless channels or other auxiliary random sources in the physical layer [8]–[16]. One noteworthy observation is that, while the work of [5]–[7] considers an asymptotic approach, in practice Alice and Bob do not usually have access to large numbers of values drawn from their random variables, but rather to only one or a few values. To address this issue, [17] shows that for such single-shot scenarios, the smooth minimum entropy provides tight upper and lower bounds on the achievable size of the secret key.

In MANETs, the lack of infrastructure, the nodes' mobility and the fact that packets are routed by nodes, instead of fixed devices, have resulted in the need for specialized routing protocols, like the ad-hoc on-demand distance vector AODV routing, or the dynamic source routing (DSR) [18]. For our secret-common-randomness-extraction purposes, DSR appears to be a good candidate, and will be the object of this paper. Indeed, for generating secret common randomness between two separated nodes in the network, they must have some shared and extractable information. Among other routing protocols in ad hoc networks, DSR has this primary feature. Namely, DSR contains two main mechanisms – Route Discovery and Route Maintenance – which work together to establish and maintain routes from senders to receivers. The protocol works with the use of explicit *source routing*, which means that the ordered list of nodes through which a packet will pass is included in the packet header. It is sets of these routing lists that we shall show how to process into secret keys shared between pairs of nodes.

Our contributions can be summarized as follows:

1) We show that the randomness inherent in an ad-hoc network can be harvested and used for establishing secret keys between pairs of nodes that participate in the routing process.
2) We provide a very practical algorithm for establishing such secret common randomness, based on the DSR protocol, and we calculate a lower bound and an upper bound on the achievable number of shared secret bits, using an adversary's beliefs.
3) We simulate a realistic ad-hoc network in OPNET Modeler, and show that within only ten minutes, thousands of secret bits can be shared between different node pairs.

The rest of this paper is organized as follows. Those parts of the DSR protocol that are essential for understanding our algorithm are examined in Section II. In Section III, we describe the system model and state our assumptions. Section IV describes our proposed key establishment algorithm. Simulation results obtained with OPNET Modeler are presented and discussed in Section V, while Section VI draws conclusions and discusses future work.

II. Dynamic Source Routing

Dynamic source routing (DSR) [18] is one of the well-established routing algorithms for ad-hoc networks. Under this protocol, when a user (the sender) decides to send a data packet to a destination, the sender must insert the *source route* in a special position of the packet's header, called the *DSR source route option*. The *source route* is an ordered list of nodes that will help relay the packet from its source to its destination. The sender transmits the packet to the first node in the *source route*. If a node receives a packet for which it is not the final destination, the node will transmit the packet to the next hop indicated by the *source route*, and this process will continue until the packet reaches its destination.

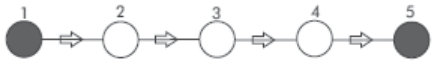

Fig. 1. Communication among node 1 and 5.

To obtain a suitable source route toward the destination, a sender first searches its own *route cache*. The *route cache* is updated every time a node learns a new valid path through the network (whether or not the node is the source or the destination for that path). If no route is found after searching the route cache, the sender initiates the *route discovery* protocol. During the route discovery, the source and destination become the *initiator* and *target*, respectively.

As a concrete example, suppose node 1 in Figure 1 wants to send packets to node 5. Initially, node 1 does not have any route toward node 5, and thus node 1 initiates a route discovery by transmitting a single special local broadcast packet called *route request*. The *route request option* is inserted in the packet's header, following the IP header. To send the route request, the source address of the IP header must be set to the address of the initiator (node 1), while the destination address of IP header must be set to the IP limited broadcast address. These fields must not be changed by the intermediate nodes processing the route request. A node initiating a new route request generates a new identification value for the route request, and places it in the ID field of the route request header. The route request header also contains the address of the initiator and that of the target. The route request ID is meant to differentiate between different requests with the same initiator and target – it should be noted here that the same request may reach an intermediate or destination node twice, over different paths. Each route request header also contains a record listing the address of each intermediate node through which this particular copy of the route request has been forwarded. In our example, the route record initially lists only the address of the initiator node 1. As the packet reaches node 2, this node inserts its own address in the packet's route record, and broadcasts it further, and so on, until the packet reaches the target node 5, at which point its route record contains a valid route (1-2-3-4-5) for transmitting data from node 1 to node 5.

As a general rule, recent route requests received at a node should be recorded in the node's *route request table* – the sufficient information for identifying each request is the tuple (initiator address, target address, route request ID). When a node receives a route request packet, several scenarios can occur. First, if the node is the target, it sends a *route reply* packet to the initiator, and saves a copy of the route (extracted from the route request route record) in a table called the *route cache*. Second, if the node has recently seen another route

request message from this same initiator, carrying the same id and target address, or if the node's own address already exists in the route record section of the route request packet (the same request reached the node a second time), this node discards the route request. Third, if the request is new, but the node is not the target, the node inserts its address in the packet's route record, and broadcasts the modified packet. Fourth, if a route exists to the target address in the node's route cache, the node sends the route reply.

In our example in Figure 1, node 5 constructs a *route reply* packet and transmits it to the initiator of the route request (node 1). The source address in the IP header of the route reply packet is set to the IP address of the sender of the route reply (node 5). In our example, node 5 is also the target. But this need not occur. Under the DSR protocol, it is possible that an intermediate node (who is not the target of the route request) already has a path to the target in its route cache. Then it is this node that transmits the route reply back to the initiator, and it is its IP address that gets inserted in the source IP address part of the route reply packet's header. The route reply packet header also contains a *route record*. This route record starts with the address of the first hop after the initiator and ends with the address of the target node (regardless of whether the node that issues the route reply is the target or not). In our example, the route record contained in the route reply packet is (2, 3, 4, 5). Including the address of the initiator node 1 in the route record would be redundant, as the address of node 1 is already included as the destination address in the IP header of the route reply packet. The combination of the route record and destination address in the IP header is the *source route* which the initiator will use for reaching its target. It is also noteworthy that network routes are not always bidirectional. That is, it may not always be possible for node 5 to send its route reply to node 1 using a route obtained by simply inverting the source route. In the more general case, node 5 has to search its own route cache for a route back to node 1. If no such path is found, node 5 should perform its own route discovery for finding a *source route* to node 1.

III. System Model

Mobile ad-hoc networks (MANETs) consist of mobile nodes communicating wirelessly with each other, without any pre-existing infrastructure. We consider a *bidirectional* MANET employing dynamic source routing (DSR), in which the nodes (corresponding to the mobile devices of the network's users) are moving in a random fashion in a predefined area. The bidirectional network assumption is usually a practical one, especially when all the nodes in the network belong to the same class of devices (e.g. smart phones).[1]

According to the route discovery protocol outlined in Section II, every single node in the network is assumed equally likely to be the initiator of a route request packet, at any given time. Furthermore, we assume that the target of any route request is uniformly distributed among the remaining nodes. Any route discovery instance will return a path through the network (the source route), of a given length. The length of a returned path is distributed according to a probability distribution that depends on all the parameters of the network. Deriving a model for this probability distribution, based on the network parameters, is outside the scope of this work. In the remainder of this paper, we shall assume that all nodes have access to such an (empirically-derived) probability distribution over the path lengths. That is, if we denote the random variable describing the length of some path r by L_r, then we assume that all the nodes have access to the prior $p(L_r = l)$, for $l = 2, 3, \ldots$. For our experiments, we run our simulation for a long time, and derive $p(L_r = l)$ by counting the paths of equal length. We also assume that *all paths of the same length are equally probable*. To express this notion, denote the random variable that samples a path (or a partial path) by R. Then we can write $p(R = r|L_r = l) = \frac{1}{N_l}$ if the length of path r is l (otherwise the probability is zero), where N_l is the total number of paths of length l. This leads to $p(R = r) = \frac{1}{N_{l_r}} p(L_r = l_r)$, where l_r is the length of path r.

Our protocol, called *KERMAN* runs by making each node collect in a table all the source routes that it is part of – recall that since the network is assumed to be bidirectional, a node can extract the route request ID, the initiator and the target from the route request packet, save them in a temporary table and if a route reply packet carrying a source route with the same initiator and target is observed within a pre-determined time interval, the node can associate the source route with the route request ID, and save both in a long-term table.

This mechanism brings about our security model. Since the common randomness established between two nodes by our algorithm consists of the source routes, it should be clear that several other nodes can be privy to this information. For instance, all the nodes included in a particular source route have full knowledge of this route. Moreover, it is likely that the route reply packet carrying a source route can be overheard by malicious eavesdroppers that are not part of the source route at all. Therefore, to achieve a level of security, two nodes will have to gather a large collection of source routes, such that *none of the other nodes that appear in any of the source routes in this collection has access to all the routes in the collection*. Unfortunately this is not enough, because it is still possible that one of the nodes, most likely a node that is part of many – though not all – routes in the collection, eavesdropped on all the remaining routes that it is not part of.

We deal with this problem by making an additional assumption: we assume that any two source routes are exchanged under independent and uniformly distributed network arrangements. That is, for the exchange (route discovery) of each source route, all the nodes in the network are distributed uniformly, and independently of other exchanges, in their predefined area. Moreover, the network remains the same for the entire duration of the route discovery and the associated data transmission. These assumptions are realistic for moderate network loads, and imply that the network nodes move around fast relative to the time between two different route

[1]It should be noted that our algorithm should work (albeit with some reduction in performance) even if the network is not bidirectional. In this case, the route request ID needs to be inserted in the route reply packet. The reduction in performance for this scenario follows from the security considerations – namely, more nodes are involved in the routing mechanism, and hence have access to the source route.

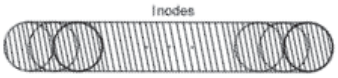

Fig. 2. The area covered by l nodes

discovery phases, but slow relative to the duration of a single communication session. This means that for any source route, the probability that any node which is *not* itself part of the route overhears the route (by overhearing a route reply or a data packet) is only a function of the network parameters. In the remainder of this section, we show how to compute the probability that an eavesdropper Eve knows a source route of which it is not part.

Denote the binary random variable encoding whether an eavesdropper Eve overhears a source route r by $K_{Eve}(r)$. Then $p(K_{Eve}(r)=1)$ depends on: (a) Eve's reception radius, (b) the total area of the network (all the places where Eve could be during the communication session corresponding to source route r), and (c) the length of the path. The computation is described in Figure 2, where it can be observed that the worst-case scenario for a path of length l is when all the l nodes are arranged in a straight line. In this case, we can use the following worst-case approximation (obtained by first calculating the area of a circular segment):

$$\begin{aligned} &p(K_{Eve}(r)=1|L_r=l) \\ &\quad=\frac{\text{Shaded area in Figure 2, where circles have radius } d_e}{\text{Total network area}} \\ &\quad=\frac{l\pi d_e^2-2(l-1)d_e^2(\frac{\pi}{3}-\frac{\sqrt{3}}{4})}{S_{total}}=\frac{d_e^2(1.91\cdot l+1.23)}{S_{total}}, \end{aligned} \quad (1)$$

where d_e is the maximum eavesdropping range (the radius of the circles in Figure 2), which is assumed the same for each of the nodes (all nodes transmit with the same power, using isotropic antennas), and S_{total} is the total area of the pre-defined location where the nodes can move.

Finally, for brevity of presentation in the current version of this work, two additional assumptions are made: the attackers are purely passive eavesdroppers (as attackers – otherwise, they are allowed to initiate well-behaved communication, just like any other node), and they do not collude. Dealing with active and colluding attackers is the subject of future work.

IV. Proposed Algorithm

In this section we introduce KERMAN, a *K*ey-*E*stablishment algorithm based on *R*andomness harvested from the source routes in a *MAN*ET employing the DSR algorithm. To establish secret common randomness between two nodes in the MANET, KERMAN uses the standard sequence of three steps outlined in Section I: advantage distillation, information reconciliation and privacy amplification.

A. Advantage Distillation

To accomplish advantage distillation, every node in the network has to maintain a new table called the *Selected Route Table*, or SRT. The SRT contains those source routes that include that node's address, and *for which the route's destination and route-reply sender do not coincide*. To demonstrate how the SRT is built, we consider the following example. Take the scenario in Figure 3, in which node 1 and 6 are the source and the destination, respectively. Since node 1 does not have any route to node 6, it generates and broadcasts a route request packet. Assume that the id of this packet is 14, which means that this is the fourteenth attempt that node 1 makes to reach node 6. Further assume that the route request first reaches node 5 over the path 1-2-3-4-5. As seen in Figure 3, node 5 will generate the route reply from its own route cache (because we assumed that node 5 already knows how to reach node 6). The transmission path of the route reply from node 5 to node 1 is the upper path in Figure 3 (that is, 5-4-3-2-1), and is consistent with a bidirectional network. Each intermediate node that receives this route reply inserts the source route in their own SRT. The SRT has three columns dubbed *RID*, *partial route* and *full route* respectively. RID is a tuple that consists *(Source IP, Destination IP, route request ID, route-reply-sender IP)*. In our scenario, nodes 1, 2, 3, 4 and 5 will all record an entry in their respective SRTs, with the RID 1-6-14-5. The intermediate nodes (2, 3 and 4) can obtain the route request ID by searching their own *route request tables* as discussed in Section II. The *partial route* field of the SRT entry identifies those other nodes that are supposed to have this particular route in their SRT – in this case, nodes 1, 2, 3, 4 and 5. The *full route* field is the entire route from source to destination, which will be used for data transmission (1,2,3,4,5,6 in this case). The SRTs of the nodes 1, 2, 3, 4 and 5 have the same following entry:

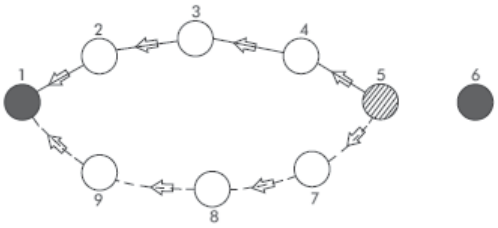

Fig. 3. Example for proposed algorithm

RID	Partial Route	Full Route
1-6-14-5	1-2-3-4-5	1-2-3-4-5-6

It should be noted that, because node 6 did not directly hear the route request from node 1, it has no way of determining the route request ID in the RID, and this is why it cannot store this entry in its SRT, although it will most likely learn the source route from the received data packets that follow the route discovery phase. Thus, although node 6 will not use this specific route for establishing a secret key with one of its peers, when discussing the security of the established secret common randomness between two other peers sharing this route, node 6 will be considered a possible eavesdropper (i.e. node 6 will be assumed to have full knowledge of the *full route*). Each full route in a nodes' SRT is only available to a limited number of nodes in the network, i.e., those nodes which are included in in the source (full) route, along with some nodes who are not part of the source route but happen to overhear the route

request and route reply exchange. The following proposition states that SRT entries are unique in the whole network. Its proof is intuitive and is available in [19].

Proposition 1: If two nodes have the same RID in their own SRTs, then the full routes associated with this RID in two SRTs are exactly the same.

B. Information Reconciliation

Information reconciliation is usually a complex process, involving techniques from channel or source coding, and displaying very restrictive lower bounds on the amount of information that needs to be transmitted over a public channel [17] – these bounds can often leave very little uncertainty for an eavesdropper. Fortunately, KERMAN is particularly well-suited for information reconciliation, and only requires minimal communication overhead. This is due to the fact that in KERMAN the common randomness is based on full routes, and each full route is uniquely identified, at both parties, by its RID, thus making reconciliation simpler.

Let us assume that two nodes –call them Alice and Bob for simplicity – realize that they share a large number of routes in their SRTs. For instance, Alice could first notice that Bob is part of a large number of partial routes in her SRT, and could ask Bob to perform information reconciliation, with the purpose of eventually generating a shared secret key. Upon Bob's acceptance, Alice sends him the list of RIDs corresponding to the partial routes in Alice's SRT that include the address of Bob. Bob can then verify whether he already has the received RIDs in his SRT, and can send back to Alice only those RIDs that he could not locate. The information reconciliation is now complete. Alice and Bob share a set of full routes, which constitute their common randomness.

There is but one caveat. As mentioned in Section IV-A, the RIDs consist of the tuples (Source IP, Destination IP, route request ID, route-reply-sender IP) corresponding to each route request/route reply pair. Moreover, it is possible that Alice and Bob are neither the source nor the destination, nor the route-reply sender. Thus, transmitting an RID in the clear, over a public channel, may expose up to five nodes of the route (source, destination, route-reply sender, Alice and Bob) to an eavesdropping adversary. Many practical solutions can be employed to limit the amount of information that the reconciliation leaks to potential eavesdroppers. As a starting point, several solutions are provided in [20].

But such solutions are outside the scope of this work. Instead, we take a different approach, and provide a lower bound and an upper bound on the total number of secret bits achievable by KERMAN, network-wide. For the lower bound, we consider the case when the RIDs are indeed transmitted in the clear, while for the upper bound, we consider the case where the RIDs are transmitted while being completely protected (by some hypothetical encryption mechanism) from any potential eavesdroppers. In both scenarios, however, we assume that every node in the network can see that Alice and Bob exchange RIDs – and thus any eavesdropper knows that the identities of Alice and Bob are part of the full routes used for secret key generation.

TABLE I
DIFFERENT GROUPS AND TYPES WHEN WE SEND RID IN CLEAR

Group	Type	Source	Destination	RREP Sender
1	1	A	B	X
	2	A	X	B
	3	X	A	B
2	4	A	X	Y
	5	X	A	Y
	6	X	Y	A
3	7	X	Y	Z

1) The Lower Bound: RIDs Transmitted in the Clear: Some information about the full routes is known to leak from the corresponding RIDs. But exactly how much information leaks is subject to the properties of the (Alice, Bob, route, RID) tuple. More precisely, these tuples can be divided into seven types, which can then be grouped into three different groups, according to their information-leakage behavior, as shown in Table I. Group 1 consists of the cases in which the RID reveals information about a single node, in addition to Alice and Bob. Groups 2 and 3 include the cases in which the RIDs leak information about two and three nodes, respectively, in addition to Alice and Bob. In Table I, A and B stand for Alice and Bob (and are interchangeable), while X and Y represent two nodes other than A and B. For example, in Group 2, type 4, Alice is the source but destination and route replier are two distinct nodes other than Bob.

2) The Upper Bound: RIDs Completely Protected: In this case, the only information that leaks to an eavesdropper in the process of information reconciliation is that the identities of Alice and Bob have to appear in every one of the full routes, the RIDs of which are being exchanged between Alice and Bob.

C. Privacy Amplification

For the purposes of this section we shall represent the full routes as sets of node identifiers, or addresses. Alice and Bob share a list of common full routes. Now Alice and Bob can construct the set $\mathcal{M} = \{m_1, m_2, \ldots, m_h\}$ where m_i (we'll call it a *trimmed route*) is produced from the full route r_i, by removing the addresses of Alice and Bob. At this point, full routes and trimmed routes are in a one-to-one correspondence. However, it is essential that the reader remembers the difference between a full route and a trimmed route.

In the next step, Alice partitions the set of trimmed routes $\mathcal{M}$ into several *disjoint* subsets $\mathcal{M}_k \subset \mathcal{M}$ of various sizes h_k, such that, for any $\mathcal{M}_k = \{m_{1,k}, m_{2,k}, \ldots, m_{h_k,k}\}$, the probability that any node in the network has knowledge of all the h_k trimmed routes is less than a small security parameter ϵ_1. This means that, with probability larger than $1 - \epsilon_1$, there exists at least one trimmed route in $\mathcal{H}$ that Eve knows nothing about – note that this is true for any identity that Eve may take (except, of course Eve cannot be Alice or Bob). It is the full route corresponding to this trimmed route (different from any node's perspective) that constitutes the randomness of the generated secret.

To extract a secret from each of the sets $\mathcal{M}_k$, Alice first represents all the *full routes* by binary strings of the same

length (according to a mapping previously agreed upon by all the nodes in the network). The length of the strings is determined as the logarithm to base two of the total number of possible full routes, in a practical scenario. For example, from our simulations, we noticed that full routes are limited to 15 nodes, which means that trimmed routes are limited to 13 nodes. In a network of 50 nodes, there are thus $\binom{48}{1}3! + \binom{48}{2}4! + \ldots + \binom{48}{13}15!$ possible full routes involving Alice and Bob, where the factorial terms account for all the possible arrangements. For example, there are $\binom{48}{1}$ trimmed routes of length 1, and their corresponding full routes have length 3 (this includes the node that defines the trimmed route, Alice and Bob), and there are $3! = 6$ possible arrangements of these three nodes. This total number of possible full routes amounts to representing each full route on 78 bits. The binary sequences representing the *full routes* corresponding to the trimmed routes in $\mathcal{M}_k$ are then XORed together. The result is inserted into a randomness extractor [21], which outputs a shorter bit string s_k – the secret. The secret s_k should satisfy the (ϵ_1, ϵ_2)-security defined below. For the sake of completeness, we first define smooth min entropy.

Definition 1 Smooth Minimum Entropy From [17]: Let X be a random variable with alphabet $\mathcal{X}$ and probability distribution of $P_X(x)$, and let $\epsilon_3 > 0$. The ϵ_3-smooth min-entropy of X is defined as

$$H_\infty^{\epsilon_3}(X) = -log \max_{Q_X \in \mathcal{B}^{\epsilon_3}(P_X)} Q_X(x) \tag{2}$$

where the maximum ranges over the ϵ_3-ball $\mathcal{B}^{\epsilon_3}(P_X)$ [17].

When $\epsilon_3 = 0$, smooth min entropy becomes min entropy.

Definition 2: In the context of a MANET, a piece of secret common randomness s_k established between two nodes Alice and Bob is called (ϵ_1, ϵ_2)-secure if, with probability larger than $1-\epsilon_1$, the secret s_k is ϵ_2-close to uniform from the perspective of any node in the network, except Alice and Bob.

It has been shown in [17] that the number of completely random bits that can be extracted from a bit sequence should be upper bounded by, but very close to, the smooth min-entropy of the sequence. Thus, for the purposes of this paper, we shall only focus on the (smooth) minimum entropy of a full route, viewed from the perspective of an eavesdropper. This minimum entropy is a good indication of the number of secret random bits that can be extracted from each set $\mathcal{M}_k$, and can be calculated according to Definition 1, where the probability distribution is that which characterizes Eve's belief about the full route. Eve's belief depends on whether the RID is sent in clear or perfectly protected.

1) The Lower Bound: RIDs Transmitted in the Clear: When the RIDs are communicated between Alice and Bob in the clear, Eve will be able to infer some information about the corresponding full routes agreed on by Alice and Bob. In addition, the very fact that Eve did not overhear the full route can also leak some information: longer routes are more likely to have been overheard by Eve. Thus, we are primarily concerned with the probability distribution $p(r|K_{Eve}(r) = 0, RID(r))$, where K_{Eve} is the binary random variable encoding whether Eve knows the full route ($K_{Eve} = 1$) or not ($K_{Eve} = 0$), and $RID(r)$ is the RID corresponding to the route r. Since we already saw that the information leaked to Eve from the RID depends on the group corresponding to the tuple (Alice, Bob, route, RID) – see Table I – and since for a specific group all routes of the same length are equally probable from Eve's perspective, we can write:

$$p(r|K_{Eve}(r) = 0, RID(r))) = \begin{cases} \dfrac{p(L_r = l_r|K_{Eve}(r) = 0, group = 1)}{\binom{N-3}{l_r-3}(l_r-2)!}, & group = 1 \\ \dfrac{p(L_r = l_r|K_{Eve}(r) = 0, group = 2)}{\binom{N-4}{l_r-4}(l_r-2)!}, & group = 2 \\ \dfrac{p(L_r = l_r|K_{Eve}(r) = 0, group = 3)}{\binom{N-5}{l_r-5}(l_r-2)!}, & group = 3 \end{cases} \tag{3}$$

where N is the total number of nodes in the network, the random variable L_r represents the length of the full route (l_r is the actual length of route r), and the denominators stand for the possible number of routes of length l_r, and belonging to group i, with $i \in \{1, 2, 3\}$. For example in the case of group 1, the number of full routes with length l_r in which Eve already knows the identities of three nodes (see Table I) is equal to $\binom{N-3}{l_r-3}(l_r-2)!$. This is because the unknown $l_r - 3$ nodes can be picked in $\binom{N-3}{l_r-3}$ ways, and then all the nodes, except source and destination, can be arranged in $(l_r - 2)!$ ways. It now remains to compute $p(L_r = l_r|K_{Eve}(r) = 0, group = 1)$. We can write:

$$p(L_r = l_r|K_{Eve}(r) = 0, group = i) = \frac{p(L_r = l_r|group = i)p(K_{Eve}(r) = 0|L_r = l_r, group = i)}{\sum_l p(L_r = l|group = i)p(K_{Eve}(r) = 0|L_r = l, group = i)},$$

where

$$p(L_r = l_r|group = i) = \frac{p(L_r = l_r)p(group = i|L_r = l_r)}{\sum_l p(L_r = l)p(group = i|L_r = l)}. \tag{4}$$

Now $p(L_r = l)$ is derived empirically from our simulation results, as explained in Section III, while $p(group = i|L_r = l)$ can be written as:

$$p(group = i|L_r = l) = \begin{cases} \dfrac{6}{(l_r)}\dfrac{1}{(l_r-1)}, & i = 1 \\ \dfrac{6}{(l_r)}\dfrac{(l_r-3)}{(l_r-1)}, & i = 2 \\ \dfrac{2(l_r-3)}{(l_r)}\dfrac{(l_r-4)}{(l_r-1)}, & i = 3. \end{cases} \tag{5}$$

To explain (5) consider, for example, $p(group = 2|L_r = l_r) = p(type = 4|L_r = l_r) + p(type = 5|L_r = l_r) + p(type = 6|L_r = l_r)$ (see Table I). The three probabilities on the right hand side are all equal. Let's now look at $p(type = 4|L_r = l_r)$. Consider a given route of length l_r, where the component nodes are indexed as 1 (source), $\ldots, l_r$ (destination), and imagine that Alice, Bob and the route-reply node (RR) pick uniformly randomly amongst these indices, with the caveat that Alice cannot be equal to Bob. Then $p(type = 4|L_r = l_r) = p(Alice = 1)p(Bob \neq RR \wedge Bob \in \{2, \ldots, l_r - 1\}) + p(Bob = 1)p(Alice \neq RR \wedge Alice \in \{2, \ldots, l_r - 1\}) = 2\frac{1}{l_r}\frac{l_r-3}{l_r-1}$.

TABLE II

NUMBER OF SUBSETS, OBTAINED BY THE NAÏVE ALGORITHM WITH $\epsilon_1 = .001$, FOR RIDS SENT IN THE CLEAR. TOTAL NETWORK-WIDE ACHIEVABLE NUMBER OF SHARED SECRET BITS, IN LAST COLUMN

No. of Subsets	1			2			3			4			
Group	1	2	3	1	2	3	1	2	3	1	2	3	B_{total}
No. of Pairs-naïve	203	125	3	31	16	1	4	2	0	3	1	0	862

Finally, whether Eve has eavesdropped a certain route or not does not depend on the roles of Alice and Bob in the path, nor on the identity of the route-reply sender. So we can write the last term of (4) as $p(K_{Eve}(r) = 0|L_r = l_r, group = i) = p(K_{Eve}(r) = 0|L_r = l_r) = 1 - p(K_{Eve}(r) = 1|L_r = l_r)$, which can be computed from (1).

2) The Upper Bound (RIDs Completely Protected): When the RID is perfectly protected, the probability of a certain route, from Eve's perspective, depends solely on its length. Since all unknown routes of a given length are equally probable from Eve's perspective, we can write

$$p(r|K_{Eve}(r) = 0)) = \frac{p(L_r = l_r|K_{Eve}(r) = 0)}{\binom{N-2}{l_r-2} l_r!}, \quad (6)$$

where the denominator represents the number of all possible routes of length l_r that contain Alice and Bob (similarly to (3)). Now we can write the right-hand of 6 side as:

$$p(L_r = l_r|K_{Eve}(r) = 0) = \frac{p(L_r = l_r)p(K_{Eve}(r) = 0|L_r = l_r)}{\sum_l p(L_r = l)p(K_{Eve}(r) = 0|L_r = l)}. \quad (7)$$

In the right-hand side of (7), $p(L_r = l_r)$ is the empirically-derived probability distribution discussed in Section III, while $p(K_{Eve}(r) = 0|L_r = l_r) = 1 - p(K_{Eve}(r) = 1|L_r = l_r)$ can be computed from (1).

3) The Partitioning Algorithm: Now the remaining question is how many subsets $\mathcal{M}_k$ we can form. To solve this problem, for any pair of nodes we organize the full set of all trimmed routes $\mathcal{M}$ as a *selection matrix*. In the *selection matrix*, a row corresponds to one of the trimmed routes in $\mathcal{M}$. A column corresponds to a node's address. There are 48 columns (one for each node in the MANET, except Alice and Bob). Each entry in the matrix is the probability that the node in the respective column knows the full route corresponding to the respective row. The selection matrix can be represented as follows:

$$\begin{array}{c} \\ m_1 \\ m_2 \\ \vdots \\ m_h \end{array} \begin{array}{c} \begin{array}{cccc} node\ 1 & node\ 2 & \dots & node\ t \end{array} \\ \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1t} \\ a_{21} & a_{22} & \dots & a_{2t} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nt} \end{pmatrix} \end{array}$$

where a_{ij} is the probability that node j knows full route i. For example, when node j is a part of the full route corresponding to the trimmed route i, then $a_{ij} = 1$. Otherwise, $a_{ij} = p(K_j(i) = 1|L_i = l_i)$, where l_r is the length of route i. The partitioning algorithm consists of constructing *distinct* sub-matrices $\mathcal{M}_k$, each consisting of h_k rows of $\mathcal{M}$, such that the product of the entries in each column of $\mathcal{M}_k$ be less than ϵ_1. We shall informally call this property ϵ_1-security, and we shall use the terms *subset* and *sub-matrix* interchangeably. An optimal partition maximizes the number of sub-matrices $\mathcal{M}_k$ with the ϵ_1-security property. In this paper we propose a naïve partitioning algorithm.

TABLE III

NUMBER OF SUBSETS, OBTAINED BY THE NAÏVE ALGORITHM WITH $\epsilon_1 = .001$, FOR PROTECTED RIDS. TOTAL NETWORK-WIDE ACHIEVABLE NUMBER OF SHARED SECRET BITS, IN LAST COLUMN

No. of Subsets	1	2	3	4	5	6	7	B_{total}
No. of Pairs-naïve	215	75	22	6	1	0	1	$4.98 \cdot 10^3$

TABLE IV

SIMULATION PARAMETERS

Simulation Parameters	Value
Network Size	100m*100m
Number of Nodes	50
Simulation Duration	600(sec)
Transmit Power(w)	.005
Packet Reception-Power Threshold(dBm)	-55
Speed(meters/seconds)	uniform(.5,1)
Packet Inter-Arrival Time(seconds)	exponential(1)

For the upper-bound scenario (perfectly protected RIDs), we build $\mathcal{M}_1$ by selecting the first row in the selection matrix, and adding the next row in the selection matrix, until the column-wise product condition holds. Then we move to the next row, and start building $\mathcal{M}_2$, and so on, until we run out of rows in $\mathcal{M}$.

For the lower-bound scenario (RIDs sent in the clear), we perform one more step: we append to each row of selection matrix a number which indicates the group of the corresponding RID. Since min-entropy for each group is different, and the number of extractable random bits is related to the min entropy, before applying the naïve algorithm, Alice and Bob should sort their routes based on the group number. That is, routes whose RIDs place them in groups with higher min-entropy come first. Note that in a subset with routes from different groups, Alice and Bob have to consider the worst-case scenario. As a concrete example, if a subset contains routes from groups 3,3,3,2,1 and group 1 has the least min entropy, the group can only produce a number of random bits equal to the min entropy of group 1. This is due to the fact that the worst-case scenario is when an eavesdropper knows all routes, except that belonging to group 1 (recall that Alice and bob do not know who the eavesdropper might be).

V. SIMULATION RESULTS

A. Secret Length and the Secret Bit Rate

The proposed protocol has been simulated in OPNET, using the parameters indicated in Table IV. This choice of parameters results in a maximum eavesdropping range of $d_e = 12$m. Each node sends packets to four random destinations.

TABLE V

PROBABILITY DISTRIBUTION OF AN UNKNOWN FULL ROUTE, FROM EVE'S PERSPECTIVE BASED ON SENDING RID TYPE

Route Length	3			4			5		
probability (Protected)	6.2E-4			9.0E-06			9.7E-08		
Group (Clear)	1	2	3	1	2	3	1	2	3
probability (Clear)	.428	0	0	.003	0.1346	0	2.2E-05	.001	.026
Route Length	6			7			8		
probability (Protected)	1.1E-09			1.1E-11			1.1E-13		
Group (Clear)	1	2	3	1	2	3	1	2	3
probability (Clear)	1.9E-07	8.4E-06	2E-04	1.5E-09	6E-08	1.8E-06	1.2E-11	4.5.5E-10	1.5E-8
Route Length	9			10			11		
probability (Protected)	1.2E-15			9.9E-18			1E-19		
Group (Clear)	1	2	3	1	2	3	1	2	3
probability (Clear)	1E-13	5.1E-12	1E-10	8.5E-16	3.7E-14	1E-12	9.5E-18	4.12E-16	1.1E-14
Route Length	12			13			14		
probability (Protected)	1E-21			1.6E-23			2E-25		
Group (Clear)	1	2	3	1	2	3	1	2	3
probability (Clear)	9E-20	3.8E-18	1.05E-16	1.05E-21	4.5E-20	1.23E-18	1.27E-23	5.4E-22	1.48E-20
Route Length	15								
probability (Protected)	2E-27								
Group (Clear)	1	2	3						
probability (Clear)	E-25	5.7E-24	1.57E-22						

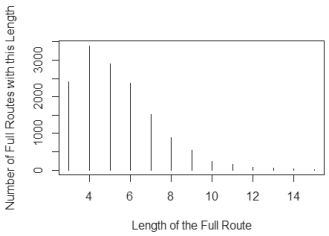

Fig. 4. Number Of Full Routes vs. Full Route Length.

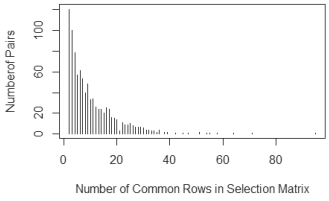

Fig. 5. Number Of Pairs vs. Number of Rows in their shared Selection matrix.

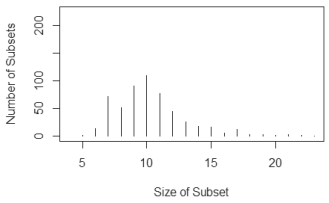

Fig. 6. Number Of subsets of a given size (number of rows), vs. Subset size, for the naïve algorithm (Clear RID) – network-wide results.

The number of full routes vs the full route length is shown in Figure 4, and the empirically-derived prior $p(L_r = l_r)$ looks similar.

As we discussed earlier, the probability distribution of the unknown full route, used in calculating the min-entropy, can be obtained from (3) (for the lower bound) or from (6) (for the upper bound). These probability distributions are given in Table V.

TABLE VI

SIZE OF MIN-ENTROPY BASED ON 3 DIFFERENT SPEEDS IN THE CASE OF CLEAR RID. SPEED 1, SPEED 2 AND SPEED 3 ARE UNIFORM (.5,1), UNIFORM (1,1.5) AND UNIFORM (1.5,2), RESPECTIVELY

	Speed 1	Speed 2	Speed 3
$H_{min}(r\|K_{Eve}(r) = 0, group = 1)$	1.22	1.237	1.267
$H_{min}(r\|K_{Eve}(r) = 0, group = 2)$	2.893	2.756	2.800
$H_{min}(r\|K_{Eve}(r) = 0, group = 3)$	5.257	4.988	5.075

It can be easily seen that when RID is sent in clear we have $H_{min}(r|K_{Eve}(r) = 0, group = 1) = -\log_2(0.428) = 1.22$, $H_{min}(r|K_{Eve}(r) = 0, group = 2) = -\log_2(0.13462) = 2.893$ and $H_{min}(r|K_{Eve}(r) = 0, group = 3) = -\log_2(0.0261) = 5.257$, while if the RID is perfectly protected we get $H_{min}(r|K_{Eve}(r) = 0)) = -\log_2(0.00062) = 10.66$.

In Figure 5 we show the number of pairs of nodes that share selection matrices, versus the number of rows in these shared matrices. Clearly, the larger the number of rows in the shared selection matrix, the higher the potential for generating more shared secret bits.

The number of subsets produced by the naïve partition algorithm for the whole network is shown in Tables II and III, for $\epsilon_1 = 10^{-3}$. We also calculate the maximum achievable total network-wide number of shared random bits (between all the possible pairs in the network), B_{total} – this is shown in the last columns of Tables II and III. For example, for $\epsilon_1 = 10^{-3}$ we have an upper bound of $B_{total} = 10.66 \cdot (215 \cdot 1 + 75 \cdot 2 + 22 \cdot 3 + 6 \cdot 4 + 1 \cdot 5 + 1 \cdot 7)$. Additionally, the numbers of subsets with a given size (number of rows) are shown for the whole network in Figure 6 and Figure 7, for the lower-bound and upper-bound scenarios, respectively.

Additionally, we evaluate the *secret bit rate, relative to transmission overhead*. Since the routing information we use for the generation of secret bits comes *free* (and is normally discarded), we normalize the number of secret bits by the number of bits transmitted for the purposes of information reconciliation, as in Section IV-B. Recall that for the reconciliation of each full route, an RID is transmitted, consisting

TABLE VII

NUMBER OF NODE PAIRS VS. NUMBER OF SUBSETS FOR THREE DIFFERENT SPEEDS BY APPLYING NAÏVE ALGORITHM WITH $\epsilon_1 = .001$, WHEN RID IS SENT IN THE CLEAR. TOTAL NETWORK-WIDE ACHIEVABLE NUMBER OF SHARED SECRET BITS, IN LAST COLUMN. SPEED 1, SPEED 2 AND SPEED 3 ARE UNIFORM (.5,1), UNIFORM (1,1.5) AND UNIFORM (1.5,2), RESPECTIVELY

No. of Subsets	1			2			3			4			5			
Group	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	B_{total}
No. of Pairs, speed 1	203	125	3	31	16	1	4	2	0	3	1	0	0	0	0	862
No. of Pairs, speed 2	209	127	1	62	34	0	13	10	0	2	4	0	0	1	0	1153
No. of Pairs, speed 3	265	147	1	51	26	0	5	8	0	0	4	0	1	1	0	1172

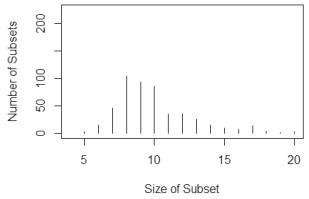

Fig. 7. Number Of subsets of a given size (number of rows), vs. Subset size, for the naïve algorithm (Protected RID) – network-wide results.

TABLE VIII

NUMBER OF NODE PAIRS VS. NUMBER OF SUBSETS FOR THREE DIFFERENT SPEEDS BY APPLYING NAÏVE ALGORITHM WITH $\epsilon_1 = .001$, WHEN RID IS PROTECTED. TOTAL NETWORK-WIDE NUMBER OF SHARED SECRET BITS, IN LAST COLUMN. SPEED 1, SPEED 2 AND SPEED 3 ARE UNIFORM (.5,1), UNIFORM (1,1.5) AND UNIFORM (1.5,2), RESPECTIVELY

No. of Subsets	1	2	3	4	5	6	7	B_{total}
No. of Pairs, speed 1	215	75	22	6	1	0	1	$4.98 \cdot 10^3$
No. of Pairs, speed 2	200	77	37	17	12	3	2	$6.63 \cdot 10^3$
No. of Pairs, speed 3	260	86	31	12	6	2	1	$6.65 \cdot 10^3$

of three node addresses and a *route-request ID*. For a network of 50 nodes, and noticing that in our simulations the route-request ID does not exceed the value of 500, the RID can be encoded on $3 \cdot \lceil \log_2(50) \rceil + 9 = 26$ bits. The additional packet header overhead is ignored here, because it is easily amortized – we could transmit many such RIDs in a single packet. The average subset size for the naïve algorithm in the case of unprotected RIDs is 9.53, and for protected RIDs it is 9.89. This implies an overhead transmission of $9.53 \cdot 26 = 238.25$ and $9.89 \cdot 26 = 257.14$ bits per subset, respectively. The *secret bit rate, relative to transmission overhead* is thus given by $1.87/238.25 = .00786$ (lower bound) and $10.66/257.14 = .0414$ (upper bound) secret bits per bit of overhead.

B. The Effects of Speed and Transmission Range

1) The Effects of Node Speed: To see the effect of the nodes' speed in the number of achieved random bits, we have simulated two additional networks, with the same parameters as those in Table IV, except with node speeds distributed uniformly over $(1, 1.5)m/s$ and over $(1.5, 2)m/s$, respectively. Based on our simulation results, the numbers of full routes of any length in the whole network, for speeds chosen as $uniform(0.5, 1)$ (the original network), $uniform(1, 1.5)$ and $uniform(1.5, 2)$ were respectively 14544, 18768 and 19900. For fully-protected RIDs, the minimum entropies (or the numbers of secret bits that can be extracted from a full route unknown by the eavesdropper), are 10.66, 10.61 and 10.67, respectively. For the case when the RIDs are sent in the clear, the min entropies corresponding to different groups are given in Table VI.

The increase in the number of route requests being generated at the whole network level with the increase of the nodes' speeds is expected, since higher node speeds result in an increased number of *broken links* – therefore, nodes have to send new discovery packets for finding new paths. On the other hand, the increase in the number of paths of a given length is roughly proportional to the original number of paths, thus leading to roughly the same minimum entropy values.

The number of achieved random bits, along with the number of subsets in the whole network are shown in Tables VII and VIII for $\epsilon_1 = 10^{-3}$. Not surprisingly, the total network-wide number of achieved shared secret bits (between any pairs of nodes) also increases with the node speeds.

2) The Effects of Transmission Range: In the following, we explore the effect of the wireless node range in the number of attained random bits. To perform this experiment, we simulate networks with the same parameters as those in Table IV, except with different wireless node ranges: 3, 6, 9, 12 and 15 meters. The number of secret random bits per subset in the case of fully-protected RIDs, for wireless ranges 3, 6, 9, 12 and 15 meters, are 8.49, 9.45, 10.28, 10.66 and 10.25 bits respectively. In the case of RIDs sent in the clear, the entropy for each group in above the ranges is shown in Table IX. The total number of secret random bits, along with the number of subsets in the whole network is shown in Tables X and XI for $\epsilon_1 = 10^{-3}$ in the case of protected and clear RID respectively.

Based on simulation results, the number of full routes of any length in the whole network for these five ranges (3, 6, 9, 12 and 15 meters) are respectively 852, 2815, 8984, 14544 and 21648. When the transmission range increases, the nodes can establish communication links more easily, causing an increase in the number of full routes, and hence in the number of shared secret bits. On the other hand, by increasing the transmission range, an eavesdropper can get information about routes more easily. It is therefore expected that the number of shared secret bits decreases as the transmission range keeps increasing beyond a certain point. For example when the range is 50m, the eavesdropper will overhear any route. Due to the large file sizes associated with the simulation of larger transmission ranges, we cannot present these results in the current version of the paper.

TABLE IX

SIZE OF MIN-ENTROPY BASED ON 4 DIFFERENT RANGES IN THE CASE OF CLEAR RID

Range	range=3	range=6	range=9	range=12	range=15
$H_{min}(r\|K_{Eve}(r)=0, group=1)$	.16	.65	1.056	1.22	1.023
$H_{min}(r\|K_{Eve}(r)=0, group=2)$	1.26	1.88	2.581	2.893	2.64
$H_{min}(r\|K_{Eve}(r)=0, group=3)$	2.58	3.988	4.751	5.257	4.74

TABLE X

NUMBER OF SUBSETS, OBTAINED BY THE NAÏVE ALGORITHM WITH $\epsilon_1 = .001$, IN THE CASE OF SENDING RID IN CLEAR FOR DIFFERENT TRANSMISSION RANGE. TOTAL NETWORK-WIDE ACHIEVABLE NUMBER OF SHARED SECRET BITS, IN LAST COLUMN

No. of Subsets	1			2			3			4			5			
Group	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	B_{total}
No. of Pairs, range=3	27	1	0	1	0	0	0	0	0	0	0	0	0	0	0	89
No. of Pairs, range=6	121	15	1	2	0	0	0	0	0	0	0	0	0	0	0	113
No. of Pairs, range=9	234	122	1	14	6	0	2	1	0	0	1	0	0	0	0	651
No. of Pairs, range=12	203	125	3	31	16	1	4	2	0	3	1	0	0	0	0	862
No. of Pairs, range=15	213	109	0	48	13	0	10	2	0	0	2	0	0	0	0	1480

TABLE XI

NUMBER OF NODE PAIRS VS. NUMBER OF SUBSETS FOR FIVE DIFFERENT RANGES BY APPLYING NAÏVE ALGORITHM. TOTAL NETWORK-WIDE NUMBER OF SHARED SECRET BITS, IN LAST COLUMN

No. of Subsets	1	2	3	4	5	6	7	B_{total}
No. of Pairs, range=3	29	1	0	0	0	0	0	263.19
No. of Pairs, range=6	134	6	0	0	0	0	0	$1.38 \cdot 10^3$
No. of Pairs, range=9	284	60	5	2	0	0	0	$4.39 \cdot 10^3$
No. of Pairs, range=12	215	75	22	6	1	0	1	$4.98 \cdot 10^3$
No. of Pairs, range=15	207	75	26	5	7	0	0	$5.02 \cdot 10^3$

VI. CONCLUSIONS AND FUTURE WORK

We have shown that the randomness inherent in an ad-hoc network can be harvested and used for establishing shared secret keys. For practical network parameters, we have demonstrated that after only ten minutes of use, thousands of shared secret bits can be established between various pairs of nodes.

The number of achievable shared secret bits can be further increased by devising a more efficient partition algorithm for the generation of full-route subsets with the ϵ_1-security property, instead of the proposed naïve algorithm used in this paper.

Future work will analyze a security model where a certain number of adversaries can collude and/or actively interfere with the protocols. In addition, although this paper focuses on the routing information circulated by DSR, other types of randomness, in more general settings, can be exploited – such as the network's connectivity or traffic load.

REFERENCES

[1] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. 22, no. 6, pp. 644–654, Nov. 1976.

[2] M. Bellare and C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm," in *Advances in Cryptology*. Berlin, Germany: Springer-Verlag, 2000, pp. 531–545.

[3] S. K. Park and K. W. Miller, "Random number generators: Good ones are hard to find," *Commun. ACM*, vol. 31, pp. 1192–1201, Oct. 1988.

[4] B. Sunar, "True random number generators for cryptography," in *Cryptographic Engineering*. New York, NY, USA: Springer, 2009, pp. 55–73.

[5] U. M. Maurer, "Secret key agreement by public discussion from common information," *IEEE Trans. Inf. Theory*, vol. 39, no. 3, pp. 733–742, May 1993.

[6] R. Ahlswede and I. Csiszár, "Common randomness in information theory and cryptography—Part I: Secret sharing," *IEEE Trans. Inf. Theory*, vol. 39, no. 4, pp. 1121–1132, Jul. 1993.

[7] R. Ahlswede and I. Csiszár, "Common randomness in information theory and cryptography—Part II: CR capacity," *IEEE Trans. Inf. Theory*, vol. 44, no. 1, pp. 225–240, Jan. 1998.

[8] J. W. Wallace and R. K. Sharma, "Automatic secret keys from reciprocal MIMO wireless channels: Measurement and analysis," *IEEE Trans. Inf. Forensics Security*, vol. 5, no. 3, pp. 381–392, Sep. 2010.

[9] M. Bloch, J. Barros, M. R. D. Rodrigues, and S. W. McLaughlin, "Wireless information-theoretic security," *IEEE Trans. Inf. Theory*, vol. 54, no. 6, pp. 2515–2534, Jun. 2008.

[10] C. Ye, S. Mathur, A. Reznik, Y. Shah, W. Trappe, and N. B. Mandayam, "Information-theoretically secret key generation for fading wireless channels," *IEEE Trans. Inf. Forensics Security*, vol. 5, no. 2, pp. 240–254, Jun. 2010.

[11] A. Agrawal, Z. Rezki, A. J. Khisti, and M. S. Alouini, "Noncoherent capacity of secret-key agreement with public discussion," *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 3, pp. 565–574, Sep. 2011.

[12] Q. Wang, H. Su, K. Ren, and K. Kim, "Fast and scalable secret key generation exploiting channel phase randomness in wireless networks," in *Proc. 30th IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Shanghai, China, Apr. 2011, pp. 1422–1430.

[13] K. Ren, H. Su, and Q. Wang, "Secret key generation exploiting channel characteristics in wireless communications," *IEEE Wireless Commun.*, vol. 18, no. 4, pp. 6–12, Aug. 2011.

[14] T.-H. Chou, S. C. Draper, and A. M. Sayeed, "Key generation using external source excitation: Capacity, reliability, and secrecy exponent," *IEEE Trans. Inf. Theory*, vol. 58, no. 4, pp. 2455–2474, Apr. 2012.

[15] C. Ye and P. Narayan, "Secret key and private key constructions for simple multiterminal source models," *IEEE Trans. Inf. Theory*, vol. 58, no. 2, pp. 639–651, Feb. 2012.

[16] A. Khisti, S. N. Diggavi, and G. W. Wornell, "Secret-key generation using correlated sources and channels," *IEEE Trans. Inf. Theory*, vol. 58, no. 2, pp. 652–670, Feb. 2012.

[17] R. Renner and S. Wolf, "Simple and tight bounds for information reconciliation and privacy amplification," in *Proc. 11th Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2005, pp. 199–216.

[18] D. A. Maltz, D. B. Johnson, and Y. Hu, "The dynamic source routing protocol (DSR) for mobile ad hoc networks for IPv4," RFC 4728, The Internet Engineering Task Force, Network Working Group, Feb. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4728.txt

[19] M. R. K. Shoja, G. T. Amariucai, S. Wei, and J. Deng. (2015). "KERMAN: A key establishment algorithm based on harvesting randomness in MANETs." [Online]. Available: http://arxiv.org/abs/1504.03744

[20] G. Brassard and L. Salvail, *Secret-Key Reconciliation by Public Discussion*. Berlin, Germany: Springer-Verlag, 1994, pp. 410–423.

[21] R. Shaltiel, "An introduction to randomness extractors," in *Automata, Languages and Programming*. Berlin, Germany: Springer, 2011, pp. 21–41.

Mohammad Reza Khalili-Shoja received the B.Sc. degree in electrical engineering from Shahed University, Tehran, Iran, in 2008, and the M.Sc. degree in electrical engineering from the AmirKabir University of Technology, Tehran, in 2011. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA. Before joining Iowa State, he was a Research Assistant with the Iran Telecommunication Research Center. His research interests are focused on security, information theory, machine learning, and wireless networks.

George Traian Amariucai received the B.Sc. and M.Sc. degrees from the Polytechnic University of Bucharest, Romania, and the Ph.D. degree from Louisiana State University, Baton Rouge, in 2009. He is currently an Adjunct Assistant Professor with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA. His research interests are focused on cyber security and its intersections with information theory, cryptography, wireless networks, social networks, and machine learning.

Shuangqing Wei received the B.E. and M.E. degrees in electrical engineering from Tsinghua University, in 1995 and 1998, respectively, and the Ph.D. degree from the University of Massachusetts, Amherst, in 2003. He started his academic career at Louisiana State University (LSU) after obtaining the Ph.D. degree. He is currently a tenured Associate Professor with the Division of Electrical and Computer Engineering, School of Electrical Engineering and Computer Science, LSU, and the Michel B. Voorhies Distinguished Professorship of Electrical Engineering. His research interests include information theory, statistical inference, communication theory, and their applications in the areas of telecommunication networks and complex systems.

Jing Deng received the B.E. and M.E. degrees in electronics engineering from Tsinghua University, Beijing, China, in 1994 and 1997, respectively, and the Ph.D. degree from the School of Electrical and Computer Engineering, Cornell University, Ithaca, NY, USA, in 2002. He visited the Department of Electrical Engineering, Princeton University, and the Department of Electrical and Computer Engineering, WINLAB, Rutgers University, in Fall 2005. He was with the Department of Computer Science, University of New Orleans, from 2004 to 2008. He served as a Research Assistant Professor with the Department of Electrical Engineering and Computer Science, Syracuse University, from 2002 to 2004. He is currently an Associate Professor with the Department of Computer Science, University of North Carolina at Greensboro, Greensboro, NC, USA. He is an Editor of the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY. He was a co-recipient of the 2013 Test of Time Award by the ACM Special Interest Group on Security, Audit, and Control. His research interests include wireless network and security, information assurance, mobile ad hoc networks, and social networks.