Title

New Non-Interactive Key Agreement and Progression (NIKAP) Protocols and Their Applications to Security in Ad Hoc Networks

Author

Garcia-Luna-Aceves, J.J.

Publication Date

2005-11-07

Peer reviewed

New Non-Interactive Key Agreement and Progression (NIKAP) Protocols and Their Applications to Security in Ad Hoc Networks[1]

Zhenjiang Li[†], J.J. Garcia-Luna-Aceves[† ‡]
[†]Department of Computer Engineering, University of California, Santa Cruz
1156 high street, Santa Cruz, CA 95064, U.S.A.
Phone:1-831-4595436, Fax: 1-831-4594829
Email: {zhjli,jj}@soe.ucsc.edu
[‡]Palo Alto Research Center (PARC), 3333 Coyote Hill Road, Palo Alto, CA 94304

Abstract

Symmetric cryptographic primitives are preferable in designing security protocols for mobile ad hoc networks (MANETs) because they are computationally affordable for resource-constrained mobile devices forming a MANET. Most proposed key-distribution and key-agreement schemes for symmetric cryptosystem assume services from on-line centralized authorities, or require the interaction between communicating parties. However, the presence of a centralized authority violates the ad hoc definition of MANETs, and interactive schemes require the routing of the ad hoc network to be established before the key agreement, which is difficult to ensure in a mobile ad hoc network (MANET).

We propose a new non-interactive key agreement and progression (NIKAP) scheme for MANETs, which does not require an on-line centralized authority, can establish and update pairwise shared keys between any two nodes in a non-interactive manner, is configurable to operate synchronously (S-NIKAP) or asynchronously (A-NIKAP), and is able to provide differentiated security services w.r.t. specified security policies. As the name implies, NIKAP is especially valuable to scenarios in which shared secret keys are desired to be computed without negotiation between nodes over insecure channels, and need to be updated frequently.

Keywords: **NIKAP, symmetric cryptosystem, self-certified key (SCK), ad hoc network**

[1]This work was supported in part by the National Science Foundation under Grant CNS-0435522, by the UCOP CLC under grant SC-05-33, by the Army Research Office under grant No. W911NF-05-1-0246, and by the Baskin Chair of Computer Engineering at University of California, Santa Cruz. Any opinions, findings, and conclusions are those of the authors and do not necessarily reflect the views of the funding agencies.

1 Introduction

An ad hoc network is a self-organizing network of mobile nodes connected by wireless links, where a prior infrastructure does not exist, and can be rapidly deployed in critical scenarios such as battle fields and rescue missions. However, the ad hoc deployment without centralized administration and the highly dynamic nature of ad hoc networks also bring up new challenges to systems built on them, amongst which security is a pressing problem.

In general, there are three cryptographic techniques that can be used to enhance the security of ad hoc networks: hash functions, symmetric cryptosystems and asymmetric (or public key) cryptosystems. An asymmetric cryptosystem is more efficient in key utilization in that the public key of a node can be used by all the other nodes, while a symmetric cryptosystem requires the existence of a shared key between any two nodes. Hash functions can be implemented quickly, and usually work together with symmetric or asymmetric algorithms to create more useful credentials, such as a digital certificate or keyed hash value.

Portable devices (or nodes) comprising a mobile ad hoc network (MANET) may have limited computational power (CPU) and battery life-time, and must share a relatively limited transmission bandwidth. Therefore, symmetric cryptosystems are preferable in ad hoc scenarios due to their computational efficiency (asymmetric algorithms usually are three or four orders of magnitude slower than the symmetric counterparts). In symmetric cryptosystems, a pairwise shared key must exist between any pair of source-target nodes before any symmetric cryptographic primitive can be performed. The key establishment problem between two network principals is well understood for conventional communication networks, and generally can be resolved by key distribution or key agreement.

The classic key distribution scheme, such as Kerberos

0-7803-9466-6/05/$20.00 ©2005 IEEE

[7], requires an on-line centralized authority (CA) to generate and distribute keys for nodes. However, this is not suitable for MANETs. In practice, the on-line CA can be unavailable to some of the nodes, or even the whole network during certain time periods due to the unpredictable state of wireless links and node mobility. Because the CA is the single point of failure, compromising the CA compromises the security of the entire system. More importantly, the Kerberos system is designed to provide authentication and key distribution services for networks structured according to the client-service model, which is not the case in the ad hoc scenario. In ad hoc networks, nodes are assumed to be willing to route packets for other nodes and behave as peers of one another, such that every node has the responsibility of a mobile router in addition to a common network user. Therefore, an ad hoc network is a peer-to-peer communication system, into which the conventional client-server model oriented, centralized key distribution protocols do not fit. Recently proposed key distribution protocols [13] for ad hoc networks replace the functionality of a CA by a subset of nodes in the network. However, this approach still relies on a (small) subset of the nodes in the system, and it is not clear that this approach can perform better than protocols that rely on a single CA, because mobile nodes need to contact multiple nodes, who can be multiple hops away, in order to obtain the desired keys.

Key agreement protocols, such as the Diffie-Hellman key exchange protocol [4] and many variations derived from it, do not need an on-line CA, and shared keys can be computed between nodes in an on-demand manner. These protocols are interactive schemes in that nodes need to exchange messages between them to establish pairwise keys, and routes must pre-exist for nodes to negotiate with each other. However, the assumption of pre-existing routes between two parties, which may be multiple hops away from each other, goes against the need to secure the routing discovery process between such nodes. Even if routes could be assumed to be pre-existing, network dynamics can tear them down in the middle of the execution of the protocol, and therefore no key may be agreed upon. Moreover, interactive key agreement protocols are not scalable in terms of communication overhead, because messages exchanged for key negotiation can consume significant CPU cycles and wireless bandwidth in a highly dynamic environment. This can be much worse if we need to update the shared keys between nodes frequently.

Motivated by the observations above and based on self-certified key (SCK) [12] cryptosystem, we propose new non-interactive key agreement and progression (NIKAP) protocols to facilitate key agreement in ad hoc networks. In NIKAP oriented protocols, pairwise keys can be computed between any two nodes in a non-interactive manner, as well as the key progression (re-keying) process. Our protocols need the aid of a centralized authority (CA) only at the initial network formation, then the CA can be entirely off-line thereafter. Therefore, single points of failure are avoided. Compared to other key distribution and agreement schemes, Our approach saves valuable energy and bandwidth in transmitting, receiving and processing messages.

This paper is organized as follows. We first review the basic idea of self-certified key (SCK) cryptosystem, which was introduced in [12]. Next we present S-NIKAP and A-NIKAP, our non-interactive key agreement and progression protocols tailored for MANETs. Then we discuss scenarios in which our NIKAP-based protocols can be applied, and summarize the results of our recent use of the NIKAP scheme to secure the routing process in MANETs. Lastly, we compare our protocols with other key distribution and establishment approaches proposed for ad hoc networks.

2 The Non-Interactive Key Agreement and Progression (NIKAP) Scheme

2.1 Basics of the self-certified key (SCK) system

In an asymmetric cryptosystem, there are two ways of ensuring the authenticity of a public key: explicit verification and implicit verification. In explicit verification, a trusted centralized authority signs a certificate that binds a public key and the identity (ID) of its owner. Then any user can verify the certificate explicitly given the public key of the centralized authority. In implicit verification, the authenticity of a public key is verified when it is used for encryption/decryption, signature verification, key exchanging or other cryptographic operations. For example, a successful verification of a signature means that the public key matches the private key used to construct this signature. Self-certified key (SCK) system follows the track of implicit verification. In what follows, we summarize the procedures used by SCK to establish and update shared pairwise keys between two communicating parties. In this case, the authenticity of a public key is verified when the shared keys computed based on it are used to encrypt and decrypt data, generate and check keyed hash values, for example.

Initialization A centralized authority (CA) Z is assumed to exist before the network formation; and Z chooses large primes p, q with $q|(p-1)$ (i.e., q is a prime factor of $p-1$), a random number $k_A \in Z_q^*$, where Z_q^* is a multiplicative subgroup with order q and generator α; then Z generates its public/private key pair (x_Z, y_Z). We also assume that the public key y_Z is known to every node in the network hereafter. To issue the private key for node A with identifier ID_A, Z computes the signature parameter $r_A = \alpha^{k_A} \ (mod\ p)$ and $s_A = x_Z \cdot h(ID_A, r_A) + k_A \ (mod\ q)$, where $h(\cdot)$ is a collision-free one-way hash function and

$(mod\ p)$ means modulo p. Node A publishes the parameter r_A (also called guarantee) together with its identifier ID_A, and keeps $x_A = s_A$ as its private key. The public key of A can be computed by any node that has y_Z, ID_A and r_A according to

$$y_A = y_Z^{h(ID_A, r_A)} \cdot r_A \ (mod\ p)$$

We denote this initial key pair as $(x_{A,0}, y_{A,0})$

User-controlled key pair progression Node A can update its public/private key pair *either synchronously or asynchronously*. In the synchronous setting, where A uses the key pair $(x_{A,t}, y_{A,t})$ in time interval $[t \cdot \Delta T, (t+1) \cdot \Delta T)$, node A can choose n random pairs $\{k_{A,t} \in Z_q^*\ ,\ r_{A,t} = \alpha^{k_{A,t}}\ (mod\ p)\}$, where $1 \leq t \leq n$, and publishes guarantees $r_{A,t}$. Then the private key of node A can progress as follows:

$$x_{A,t} = x_{A,0} \cdot h(ID_A, r_{A,t}) + k_{A,t}\ (mod\ q)$$

The corresponding public keys are computed according to

$$y_{A,t} = y_{A,0}^{h(ID_A, r_{A,t})} \cdot r_{A,t}\ (mod\ p) \quad (1)$$

Non-interactive pairwise key agreement and progression Pairwise shared keys between any two nodes A and B can also be computed and updated synchronously or asynchronously as follows.

- **Node A:**
  $x_{A,t} = x_{A,0} \cdot h(ID_A, r_{A,t}) + k_{A,t}$
  $y_{B,t} = y_{B,0}^{h(ID_B, r_{B,t})} \cdot r_{B,t}\ (mod\ p)$
  $K_{A,t} = y_{B,t}^{x_{A,t}}\ (mod\ p)$
  $K_t = h(K_{A,t})$

- **Node B:**
  $x_{B,t} = x_{B,0} \cdot h(ID_B, r_{B,t}) + k_{B,t}$
  $y_{A,t} = y_{A,0}^{h(ID_A, r_{A,t})} \cdot r_{A,t}\ (mod\ p)$
  $K_{B,t} = y_{A,t}^{x_{B,t}}\ (mod\ p)$
  $K_t = h(K_{B,t})$

The pairwise shared keys obtained by node A and node B are equal because
$h(K_{A,t}) = h(y_{B,t}^{x_{A,t}}\ (mod\ p)) = h(\alpha^{x_{A,t}x_{B,t}}\ (mod\ p)) = h(y_{A,t}^{x_{B,t}}\ (mod\ p)) = h(K_{B,t})$.

Two features of SCK should be pointed out.
• Firstly, given N nodes in the network, assume that their IDs are globally known, in order to distribute their public keys, N guarantees are distributed, instead of N traditional certificates. The advantage is that, unlike a certificate based approach, these N guarantees can be published and need not to be certified (signed) by any centralized authority. This means that we can derive the public key of any node, and update the public/private key pair between any two nodes without the aid of an on-line CA (access to CA is only required at the initial network formation, as described above).

• Secondly, given that N guarantees are already distributed to all nodes in the network, and the public key of the CA is known to everyone, then any two nodes can establish a pairwise shared key, also the updated keys thereafter, in a non-interactive manner. This means that no further negotiation message is needed for key agreement and progression, such that *zero* communication overhead for key negotiation is achieved.

2.2 S-NIKAP and A-NIKAP protocols based on self-certified key (SCK)

SCK is particularly attractive for the establishment of secure protocols for MANETs, because it enables a non-interactive key agreement and progression (NIKAP) scheme for MANETs. However, the basic primitives used by SCK to establish pairwise keys between nodes cannot be applied directly to ad hoc scenarios. In this section, we present two protocols which implement NIKAP to facilitate security mechanisms using symmetric cryptographic primitives, and allow NIKAP to be configurable if time synchronization is available to mobile nodes in the network. The protocols we introduce are synchronized NIKAP (S-NIKAP) protocol and asynchronized NIKAP (A-NIKAP) protocol, respectively.

In S-NIKAP, two nodes negotiate and update the shared keys between them periodically according to the current time instant and the specified security policy. Processes or applications of higher security concern can perform the re-keying (key progression) operation at a high frequency, and those of lower security concern at a low frequency, accordingly. Therefore, communication principals in the network can be distinguished based on different security policies, such as roles, service types or the sensitivity of data etc. As a result, differentiated security services can be achieved by specifying high-to-low re-keying frequencies that correspond to high-to-low security levels. The main limitations of S-NIKAP are the prerequisite of time synchronization and the periodical re-keying at a fixed frequency. Though there exist devices or protocols providing time synchronization for ad hoc networks, it is still not clear if the desired performance can be achieved in such a highly dynamic and unpredictable environment. Another drawback of S-NIKAP is that the shared keys are updated independently of whether or not communication between nodes takes place. Therefore, local CPU cycles is wasted if the newly generated keys are not used within its life-cycle.

It follows naturally that an asynchronous version of

Algorithm 1 Protocol S-NIKAP (for any node A)

1: Node initialization:
Retrieve the CA's public key y_Z, initial private key $x_{A,0}$, initial guarantee $r_{A,0}$ and key progression interval ΔT
2: Guarantees distribution:
Broadcast ID_A and randomly selected guarantees $r_{A,t}$ where $1 \leq t \leq n$. ($r_{A,t}$ and ID_A can be broadcast over insecure channel)
3: Pairwise keys agreement and progression:
To communicate with node B in time interval $[T_0 + t \cdot \Delta T, T_0 + (t+1) \cdot \Delta T)$, first update the key shared with B to K_t, according to the key progression described in Section 2.1.

NIKAP is desired in cases in which time synchronization is not achievable or portable nodes cannot afford the cost of key progression at high frequencies. Asynchronous NIKAP (A-NIKAP) has the same non-interactive re-keying capability as S-NIKAP does, but requires no time synchronization service from the underlying network. Instead, A-NIKAP uses a pseudo-random bit stream to *synchronize* the re-keying process between nodes, of which "1" invokes new key progression while "0" keeps two nodes using the current shared key between them. According to SCK, an initial shared key can be non-interactively established. Therefore, the pseudo-random bits stream can be generated, encrypted (under the initial key), and securely established between the two nodes. If the same pseudo-random number generator is used by both ends, to save the bandwidth, only a common seed needs to be exchanged. The progression strategy in A-NIKAP can be specified as per-session based, fixed number of sessions based or fixed number of packets sent based etc., according to the given security policies. If the *bit-synchronization* is lost, nodes need to re-establish a new pseudo-random bits stream (by using the last shared key working between them, or simply start over). If we count one bit in the random bits stream equal to one time interval used in S-NIKAP, A-NIKAP incurs half of the local CPU cycles than S-NIKAP does, provided that the bits stream is perfectly randomized.

Algorithm 1 and 2 define S-NIKAP and A-NIKAP protocols, respectively.

Algorithm 2 Protocol A-NIKAP (for any node A)

1: Node initialization:
Retrieve the CA's public key y_Z, initial private key $x_{A,0}$, initial guarantee $r_{A,0}$ and key progression interval ΔT
2: Guarantees distribution:
Broadcast ID_A and randomly selected guarantees $r_{A,t}$ where $1 \leq t \leq n$. ($r_{A,t}$ and ID_A can be broadcast over insecure channel)
3: Random bits stream generation and exchange:
To communicate with node B, first generate a random bits stream $BITS_A$ and send to B as follows:
$A \Rightarrow B : \{ID_A, ID_B, BITS_A, hash(ID_A, ID_B, BITS_A, K_{A,0})\}_{K_{A,0}}$
Where the $hash$ value is computed for node B to verify the integrity of $BITS_A$
4: Bit-Controlled key progression:
while $BITS_A$ is not empty **do**
  if new session **then** ▷ *Or other triggering events*
    $flag \leftarrow pop(BITS_A)$
    if flag = 1 **then**
      update the shared key to K_t
    else
      keep using the current key K_{t-1}
    end if
  end if
end while

3 Applications of NIKAP

The non-interactive progression capability of NIKAP makes it very attractive in applications in which shared keys need to be established without negotiation through insecure channels, or need to be updated frequently. Such scenarios include secure ad hoc routing, peer-to-peer communication in combat fields and surveillance systems.

Shared keys are needed between nodes to secure the routing discovery process in MANETs, and interactive key agreement protocols are not suitable for this application, because the topology and routes in a MANET are unknown when it is first deployed. Consequently, given that there can be no pre-existing secure channels for the interaction between any pair of nodes, a common broadcast channel must be used for key exchange, which can be exploited by malicious users. In addition, requiring the collaboration among nodes to establish shared keys while they are establishing routes to one another cannot be done efficiently. The non-interactive nature of NIKAP allows nodes to secure the routing process, without incurring undue overhead.

NIKAP can be used here to provide differentiated security services in MANETs. To achieve better security, the keys shared between nodes can be updated regularly, and the keys used between different nodes can be re-keyed at different frequencies based on different security policies, such as privilege rankings, roles and location of the nodes.

Surveillance systems are often used to gather and upload critical data periodically to a command center from monitoring nodes. The topology of a surveillance system is relatively fixed compared with that of a mobile network, which exposes it to high possibility of being identified and

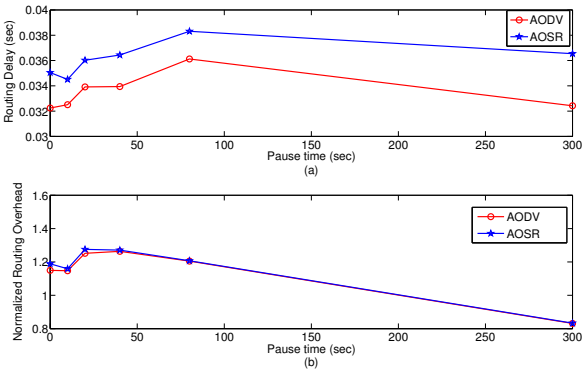

Figure 1. NIKAP and secure ad hoc routing

attacked. Therefore, keys used between the command center and each monitoring node have to be updated regularly. Moreover, pairwise key scheme is also preferable to group key scheme, in order to confine the damage caused by key divulgence. In such a case, S-NIKAP can be a good candidate for key establishment because of its periodical key progression capability.

4 Ongoing Work

In our recent work on secure ad hoc routing [9, 8], we use NIKAP to establish a pairwise shared key between each pair of nodes, and hash values keyed with them to validate the paths discovered. We use S-NIKAP in our simulation study for convenience, given that the ns2 simulator [10] provides the time synchronization. Figure 1 shows the performance comparison between our secure routing protocol (AOSR) using S-NIKAP and the ad hoc on-demand distance vector routing protocol AODV [11]. As we can see, when there are no attacks in the network, the normalized routing overhead of AOSR, defined as the number of routing packets over the number of data packets, is almost the same as that of AODV, as shown in Figure 1 (b). The reason is that, in S-NIKAP (or A-NIKAP), shared keys are established without further negotiation between nodes, and no on-line CA is needed either. Therefore, no negotiation message is exchanged between nodes, or between nodes and the CA.

In our simulation, the key progression interval is set to five seconds, and in practice, this is adjustable according to the processing power of mobile nodes, or the given security policy. Because keys shared between nodes need to be updated at a fixed frequency, we expect that the time it takes for AOSR to find routes should be longer than that of AODV. Fortunately, as shown in Figure 1 (a), the average routing delay caused by key progression, measured over all nodes, is only $2 \sim 4ms$ more than that of AODV, which is an acceptable increase of $5 \sim 8\%$. This indicates that NIKAP efficiently supports the security mechanisms used by the routing discovery process of AOSR.

5 Related Work

Existing key distribution protocols for ad hoc networks generally assume the existence of an on-line centralized authority (CA). To alleviate the risk caused by the single point of failure, threshold cryptography based protocols replace the CA by a subset of nodes which share and provide the functionality of the CA contributorily [13]. However, this approach cannot completely eliminate the reliance on an on-line CA, such that the CA is still of major interest to attackers. What can be worse is that the alternative of using multiple mobile mini-CAs requires nodes to contact up to certain number of mini-CAs before they can obtain the desired keys. Therefore, in highly dynamic scenarios such as ad hoc networks, the *responsiveness* of multiple mini-CAs schemes be worse than that of schemes based on a single CA. Key distribution protocols using ID-based cryptography [2], or the combination of threshold and ID-based cryptography [6], have the same advantage as SCK because IDs (publishable) are used to obtain the corresponding public keys of nodes, instead of using a certificate to bind the ID and its public key. However, on-line CA services must exist for these protocols to work correctly, which suffers the same limitations as that of protocols based on threshold cryptography.

Another scheme of key agreement for ad hoc networks is to combine threshold secret sharing and probabilistic key sharing[14]. The basic idea behind this is to split the shared secret between a source-target pair into several pieces, then communicate them towards the target in such a way that the target node has a high probability to recover the splitted secret based on received pieces. However, this requires that routes pre-exist between the source and the target nodes, which can be many hops away. Moreover, the overhead incurred by communicating secret pieces towards target can be high due to network dynamics, and sufficient number of secret pieces may not arrive at the target, which then cannot recover the original secret.

Group key agreement protocols [1, 5] for ad hoc networks are different from S-NIKAP and A-NIKAP. In group key agreement, a shared key needs to be established amongst all possible nodes belonging to a multicast or many-to-many-cast group, while S-NIKAP and A-NIKAP only consider the key agreement between two nodes. The storage complexity of a system using group keys is obviously less than that of a system using pairwise keys. However, in group communication, the cost of re-keying operation, caused by nodes leaving or joining, network partition and merging can be very high. The reason is that, whenever the group membership changes, a new group key must

be computed for all group members, otherwise the group communication becomes insecure due to possible keys divulgence. Another drawback of a system using group keys is that the compromise of a group key can put all group members under attacks, while the compromise of a pairwise key only affects the security of the pair of nodes using this key. In practice, whether to use a pairwise key scheme or a group key scheme depends on the application scenario and the security requirements being concerned with.

Last but not least, the key agreement protocols summarized above need nodal interaction. There exist a few protocols that can establish pairwise keys between nodes non-interactively based on either matrix threshold key predistribution (MTKP) or polynomial threshold key predistribution (PTKP) [3], but none of them supports non-interactive key progression.

6 Conclusion

We present two new protocols: S-NIKAP and A-NIKAP, in which pairwise key agreement can be achieved non-interactively, so is the succeeding key progression (re-keying) process. Our protocols save valuable energy and bandwidth in communicating and processing messages, and especially fit in with scenarios in which frequent key re-keying is needed. Our scheme needs the aid of a centralized authority only at the initial network formation, which is also better than other approaches depending on on-line CA services. Our work using NIKAP for secure ad hoc routing shows that NIKAP bootstraps key establishment in ad hoc networks efficiently, and is promising for other resource-constrained ad hoc scenarios where frequent and non-interactive key re-keying is desired.

References

[1] Y. Amir, Y. Kim, C. Nita-Rotaru, and G. Tzudik. On the Performance of Group Key Agreement Protocols. In *Proceedings of the 22nd IEEE International Conference on Distributed Computing Systems (ICDCS), Vienna, Austria*, July 2-5, 2002.

[2] D. Boneh and M. Franklin. Identity Based Encryption from the Weil Pairing. In *Proceedings of Crypto '2001, Lecture Notes in Computer Science, Vol. 2139, Springer-Verlag*, pages 213–229, 2001.

[3] C. Castelluccia, N. Saxena, and J. H. Yi. Self-Configurable Key Pre-distribution in Mobile Ad-Hoc Networks. In *Proceedings of IFIP Networking Conference, LNCS 3462, pp.1083-1095, Waterloo, Canada*, May 2005.

[4] W. Diffie and E. Hellman. New Directions in Cryptography. *IEEE Tran. Inform. Theory*, 22:644–654, Nov. 1976.

[5] A. P. H. Chan and D. Song. Random Key Predistribution Schemes for Sensor Network. In *Proceedings of IEEE Symposium on Research in Security and Privacy*, pages 197–213, 2003.

[6] A. Khalili, J. Katz, and W. Arbaugh. Towards Secure Key Distribution in Truly Ad-Hoc Networks. In *Proceedings of IEEE Workshop on Security and Assurance in Ad hoc Networks, Orlando, FL*, Jan. 28, 2003.

[7] J. kohl and B. Neuman. The Kerberos Network Authentication Service (V5). *RFC 1510*, September, 1993.

[8] Z. Li and J. Garcia-Luna-Aceves. Enhancing the Security of On-demand Routing in Ad Hoc Networks. In *Proceedings of the 4th International Conference on AD-HOC Networks and Wireless (AdhocNow'2005), Cancun, Mexico*, October 6-8, 2005.

[9] Z. Li and J. Garcia-Luna-Aceves. A Secure Reactive Ad Hoc Routing Protocol. In *Proceedings of the 3rd IEEE Latin American Network Operations and Management Symposium (LANOMS' 2003), Iguassu Falls, Brazil*, September 4-6, 2003.

[10] NS2. the Network Simulator, http://www.isi.edu/nsnam/ns/.

[11] C. E. Perkins and E. M. Royer. Ad Hoc On Demand Distance Vector Routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA*, pages 90–100, February 1999.

[12] H. Petersen and P. Horster. Self-Certified Keys - Concepts and Applications. In *Proceedings of the 3rd Conference of Communications and Multimedia Security, Athens*, September 22-23, 1997.

[13] L. Zhou and Z. Haas. Securing Ad Hoc Networks. *IEEE Network, Special Issue on Network Security*, 13(6):24–30, 1999.

[14] S. Zhu, S.Xu, S. Setia, and S. Jajodia. Establishing Pairwise Keys for Secure Communication in Ad Hoc Networks: A Probabilistic Approach. In *Proceedings of the 11th IEEE International Conference on Network Protocols*, page 326, Washington, DC, USA, 2003. IEEE Computer Society.