Distributed Certificate Authority in Cluster-based Ad hoc networks

Ghassan Chaddoud[†], Keith Martin[‡]

[†]Department of Scientific Services, Atomic Energy Commission of Syria
Damascus, P.O.Box 6091, Syria, gchaddoud@aec.org.sy
[‡]Information Security Group, Royal Holloway, University of London
Surrey TW20 0EX, U.K, keith.martin@rhul.ac.uk

Abstract

The need to secure communication in ad hoc network is extremely challenging because of the dynamic nature of the network and the lack of centralized management. This makes public key cryptographic services particularly difficult to support. We propose a distributed certificate authority intended for deployment in an NTDR cluster-based architecture. We also outline procedures for maintaining this distributed certificate authority amongst a highly dynamic membership of shareholding nodes.

1 Introduction

The increasing interest in ad hoc networks has made their security a real concern. As a result, ad hoc network security has been subject to extensive recent study. While much attention has been spent looking at security of routing protocols in ad hoc networks, for example [1], [6], [7], [12], [21], [15], it is equally important to secure communications in ad hoc networks [20], [19], [13]. In order to employ security mechanisms that are based on public key technology, it is necessary to establish the supporting key management infrastructure, which is normally based around the concept of a *certificate authority* (CA).

However, security in mobile ad hoc networks is particularly challenging for many reasons [10], including:

1. the dynamically changing topology;
2. the sporadic nature of connectivity;
3. the vulnerability of links;
4. the limited physical protection of nodes;
5. the lack of centralized monitoring or management points.

The latter is arguably the most crucial for offering public key-based security services because, in the absence of a global centralized and trusted authority, establishing a CA can be highly problematic,if not impossible.

An attractive idea is thus to distribute a CA's functionality amongst ad hoc network nodes. A *Distributed Certificate Authority (DCA)* is realized through the distribution of the CA's private key to a number of special *shareholding* DCA nodes. When CA-related operations are required, such as issuing or signing a certificate, checking public keys, or revoking certificates, a threshold of available shareholding DCA nodes should participate in the operation.

There has been relatively little work to date on designing distributed CA services (we will discuss related work in Section 4). In this paper we present a new model for a distributed certificate authority in NTDR (Near-Term Digital Radio [22]) cluster-based ad hoc networks. The DCA's private key is never known by any single node, either during setup or during certificate authority-related operations. Our DCA is designed to complement the NTDR security framework proposed in [19].

The remainder of the paper is organised as follows. In Section 2 we introduce NTDR networks and briefly outline the security architecture proposed in [19] to enable security services. Section 3 we identify desirable design properties and present our solution. In Section 4 we compare this solution with related work.

2 NTDR Ad hoc Networks

This section presents a brief overview of NTDR and a proposed security architecture.

2.1 Overview

A *Near-Term Digital Radio* (NTDR) ad hoc network [22] is a cluster-based control structure composed of organizational *clusters*, each containing a *clusterhead*. The clusterheads are linked together to form the network backbone. Clusters are formed of mobile nodes within one communication hop from the clusterhead. Inter-cluster communication is restricted to clusterheads only. Inside a cluster, nodes can communicate directly if they are within one hop of one other, otherwise all communications must pass through the clusterhead. A clusterhead thus functions as a gateway and the clusterheads collectively share responsibility for maintaining the routing backbone. This requires clusterheads to constantly monitor and distribute amongst themselves information about changes that occur on the backbone.

Since clusterheads represent potential points of failure in network operations, nodes adopt an aggressive mode in maintaining the connectivity of the backbone. To facilitate this, there is a process by which any node can quickly become a clusterhead in the event of a connectivity problem. To this end, each node keeps track of its neighbors by broadcasting periodic *beacon messages*. A beacon message contains the sender's *medium access control* (MAC) address and the lowest-numbered MAC address amongst all nodes reachable from the issuing node. In addition, a clusterhead's beacon message contains, amongst other things, the clusterhead's organizational affiliation and a list of cluster members. Nodes receiving a clusterhead's beacon message can determine whether to affiliate with the clusterhead or not.

Keeping in mind power constraints, two different frequencies are used for transmission. The strongest is used by clusterheads for inter-clusterhead communication over the backbone. The second weaker frequency is used for intra-cluster communication. Clusterheads are thus requird to use both frequencies.

A node seeking cluster affiliation is more likely to choose clusters where the signal from the clusterhead is transmitted at low power but received at high strength, the resulting cluster size is relatively small and the current clusterhead belongs to an organization recognized by the node. If the current clusterhead approves the new node affiliation, the clusterhead updates the cluster membership list and broadcasts this to the other clusterheads. This alerts all clusterheads to the node's new affiliation. A node remains affiliated to a cluster until the clusterhead relinquishes its role or technical communication problems, in particular noisy or deformed signals, occur.

2.2 Security framework for NTDR

In [19] a security framework is proposed that identifies threats related to communications in NTDR ad hoc networks and designs security services and mechanisms for thwarting such threats. In this framework, clusterheads play a key role in carrying out security related functions. More specifically, a clusterhead authenticates and grants affiliation status to nodes. Moreover, the clusterhead is responsible for managing inter-cluster and intra-cluster pair-wise and group-wise cryptographic keys, which are used to provide confidentiality, integrity, and authentication services.

Within a cluster, pairwise keys are used to secure unicast communication between nodes. Group-wise keys are used to secure group (intra-cluster) communication. Similarly, inter-cluster communication is protected by the use of pairwise and group-wise keys at the network backbone level, where in this case the group-wise key is shared by all clusterheads.

Inter-cluster keys are managed by the clusterheads using pairwise secret master keys shared with cluster nodes. These pairwise keys are established during node affiliation, based on public key techniques. The framework assumes that, prior to joining the NTDR network, each user gets a unique *off-line certificate*) along with any further high-level certificates that are required to verify off-line certificates of other nodes in the network (for example, high level certificates of other organizations managing nodes in the NTDR network) . While these off-line certificates can be used for a short period of time, they cannot be used to support online CA services for network nodes, due to the fact that the NTDR network cannot support an online CA. This problem motivates the need for the establishment of a DCA to provide online key management services to the NTDR nodes.

3 A Distributed Certificate Authority for NTDR

We first identify some desirable design properties and then propose a DCA intended to be compatible with them. Our DCA is based on the concept of a *threshold scheme* [4], [18], which is a means of distributing secret *shares* of a secret in such a way that a threshold k of shares must be pooled before the secret can be recovered. We will distribute shares of the DCA private key amongst shareholding DCA nodes using a special type of threshold scheme that requires k shareholding DCA nodes to collaborate on any operation requiring application of the DCA private key.

3.1 Design goals

Desirable design properties for a scalable DCA are as follows:

- *Availability* Nodes may seek a DCA's services at any time and require a response within a reasonable delay period. However, a solution must take into account the dynamic nature of an ad hoc environment. It can be expected that not every shareholding DCA node is reachable at any given time and, further, that the collection of shareholding DCA nodes varies over time. We thus require protocols to enable shareholding DCA nodes to leave and join the DCA.
- *Security* Since nodes may fall victim to different types of attack (for instance, capture), no important system secret values are to be trusted to any single node in the network. Thus, for example, DCA key pairs must be generated in a distributed way and the DCA private key should be usable without any single shareholding DCA node being able to reconstruct it. In addition, a key refresh protocol is required to ensure that the lifetimes of critical keys are restricted.
- *Reliability* Wherever possible, the system should avoid relying solely on the underlying communication network, since channels or nodes could be compromised. Where possible, measures should be taken to improve system robustness.
- *Efficiency* As nodes are power-limited and communication bandwidth is relatively low, protocols should attempt to minimize computations, connections and the amount of data transmitted between nodes.

3.2 Network model

We now review the network model in which this scheme operates. Although we saw in Section 2.1 that in an ideal NTDR network all entities can communicate through an intergrated network based on routing through the clusterheads, it is important to recognize that in reality communications are not reliable. Any entity can become offline or unreachable at any time, communications are potentially insecure and error-prone, and nodes can not be trusted to always be cooperative (sometimes referred to as *node selfishness* [13]).

We assume that the communication model is *partially synchronous*, meaning that messages are assumed to be delivered within a fixed bound of time. After a specified period, a message is considered lost and treated as a sender failure. Where possible we will adopt fault tolerant protocols.

3.3 DCA security services

The purpose of having a DCA in an NTDR network is to make use of public key-based cryptographic services. These services include support for authentication, integrity, confidentiality, access control and non-repudiation. These are required between nodes, between nodes and clusterheads, and between clusterheads. Public key techniques are attractive because they enable nodes to establish secure links without having prior relationships. However public key techniques are generally computationally intensive and so the application of public key cryptography in the NTDR security framework is largely restricted to initial node authentication and key establishment processes.

Public key cryptography involves two related keys, one of which is private and the other public. It is essential that the authenticity and validity of public keys is maintained, and the normal means of doing so is for a trusted entity (in our case the DCA) to issue a *public key certificate* attesting to this information. The DCA provides all public key certificate services, in particular issuing, revocation, renewing, and verification of certificates. When a new node first joins the NTDR network, it presents its offline certificate to the clusterhead with which it intends to affiliate. It should then seek an *on-line certificate* from the DCA, which is then used as the working certificate in the NTDR network.

3.4 DCA architecture

It is entirely natural that we propose that our DCA is distributed amongst clusterheads, which become the shareholding DCA nodes. This is because clusterheads hold positions of responsibility in the NTDR network and are in direct communication with one another, making them the most appropriate nodes to fulfill this role. The DCA private key must therefore be distributed and maintained amongst the clusterheads. When a new clusterhead joins the backbone they need to be issued with a share of the DCA private key. When a node seeks a DCA services (such as a certificate renewal), the node first contacts their clusterhead who then takes up the request with the other clusterheads.

We will define our DCA by specifying the following DCA operations:

- system setup or bootstrapping,
- applying a DCA private key,
- joining a new clusterhead,
- evicting an existing clusterhead,
- refreshing clusterhead shares.

3.5 Public parameters

Throughout this paper, p and q are large primes such that q divides $p-1$, and g is a generator of the subgroup G_q of Z_p^* of order q. The values p, q and g are public system parameters. In addition, let h be a hash function whose range is $\{1,..,q-1\}$.

3.6 Bootstrapping

Let H be the initial set of clusterheads at system setup time, $|H| = n$, and k be the required threshold of co-operation between clusterheads. In order to establish a (k,n) threshold sharing of a private key, we require that all clusterheads participate in the construction of the shared key. This participation takes place as part of the construction of the NTDR backbone. We use the following *Distributed Key Generation* (DKG) algorithm proposed in [17]:

1. Each clusterhead CH_i chooses s_i in Z_p and calculates $y_i = g^{s_i}\ mod\, p$.
2. CH_i creates a (k,n) threshold sharing of the secret value s_i by generating a polynomial function $f_i(z) = \sum_{l=0}^{k-1} a_{i,l}x^l$ of degree at most $k-1$ with $f_i(0) = s_i\ mod\, p$.
3. CH_i uses a secure unicast channel to distributes the subshare $f_i(j)$ to CH_j (this means that CH_i needs $(n-1)$ secure unicast channels, one to each of the other clusterheads).
4. CH_i broadcasts the values $y_{i,l} = g^{a_{i,l}}, (l \in \{0,..,k-1\})$. Theses values will be used to verify the consistency of the subshares sent by CH_i. Let $A_l = \prod_{i\in H} y_{i,l}$, where $l \in \{0,..,k-1\}$.
5. Each CH_j verifies that the subshare $f_i(j)$ received from CH_i is valid by checking that $g^{f_i(j)} = \prod_{l=0}^{k-1}(y_{i,l})^{j^l}$. If this equality holds then the value received from CH_i is correct. Otherwise CH_j broadcasts a warning to the other clusterheads that an inconsistent subshare has been received from CH_i. If at least k warnings are issued concerning CH_i then CH_i must be isolated (otherwise we label CH_i as *consistent*).
6. Let H_1 be the set of consistent clusterheads at the end of the last stage. Each CH_j in H_1 computes $x_j = \sum_{i\in H_1} f_i(j)$.

At the end of the above protocol each consistent clusterhead CH_j holds a share x_j of the DCA private key $SK = \sum_{j\in H_1} f_j(0)$. The DCA public key is given by $PK = \prod_{j\in H_1} y_j$ and can be computed from the broadcasts exchanged in Step 4) above.

Note that the integrity of all broadcast messages in the bootstrap procedure can be secured by digitally signing exchanged messages using the private key associated with the offline certificate issued to each node. A full analysis of the security of this protocol can be found in [17].

3.7 Applying a DCA private key

We now demonstrate how a DCA private key can be applied to deliver a DCA security service. Recall that no single clusterhead knows the DCA private key and it must not be constructed during any application. Consider the case of a node who wishes the DCA to digitally sign a request REQ. When the node's clusterhead receives the request it forwards it to the backbone. Any other clusterhead that receives the request uses his share of SK to sign the request and produces a *signature share*, before sending it back to the requesting node. Once the node has verified k signature shares it can use them to construct the DCA signature on REQ. This process can be realized by using a *threshold signature* scheme.

We first present a variant of the *digital signature standard* (DSS), proposed in [16]:

- Let $x \in Z_q$ be the private key. Let $y = g^x mod\, p$. The public key is then given by (p, q, g, y);
- To sign message M, first compute $m = h(M)$, generate a random number $e \in Z_q$ and then compute:

  1. $\gamma = (g^e \; mod\, p) \; mod\, q$;
  2. $\delta = \gamma x + me \; mod\, q$;

  The signature on message M is given by (γ, δ);
- To verify (γ, δ), check that:

$$\gamma = (g^{\delta/m} y^{-\gamma/m} mod\, p)(mod\, q).$$

We adopt the following (k, n) threshold signature scheme, which was proposed in [16] and is based on the preceding variant of DSS. Let $H_2 \subseteq H_1$ (where $|C_1| \geq k$) be the set of clusterheads available to assist in signing request REQ.

1. We first need to generate a random value e in a distributed way. The clusterheads in H_2 thus run an instance of the DKG algorithm (Section 3.6). Assuming that there are sufficient consistent clusterheads, the result is a subset H_3 of consistent clusterheads such that each $CH_i \in H_3$ has a share e_i of a random value e and there exist the following public values: $\tau = g^e \; mod\, p$, $\gamma = \tau \; mod\; q$, and $B_l = \prod_{i \in H_3} g^{b_{i,l}}$, where $l \in \{0, .., k-1\}$ and $b_{i,l}$ are CH_i's polynomial coefficients. Note that e is not revealed to any clusterhead during this process.
2. Each CH_i in H_3 computes $\delta_i = \gamma x_i + h(REQ) e_i \; mod\, q$ and sends it to the requesting node. Note that $|H_3|$ must be bigger than k
3. Upon receiving each δ_l, the requesting node checks the consistency of the value by verifying that $g^{\delta_l} = (y \prod_{j=1}^{k-1} (A_j)^{l^j})^{\gamma} (\tau \prod_{j=1}^{k-1} (B_j)^{l^j})^{h(REQ)}$
4. The requesting node computes δ by applying Lagrange Formula to $\{\delta_i\}$ as follows $\delta = \sum_{j=1}^{k} \prod_{d \neq j} \frac{i_d}{i_d - i_j} \delta_{i_j}$ for any $CH_{i_1}, ..., CH_{i_k} \in H_3$

Note that the DKG of parameter e is independent of REQ, so could be done prior to the reception of the request during idle time on the clusterhead backbone. Further, the integrity of each message sent in Step 2) can be secured by using digital signature.

3.8 DCA Clusterhead join

In this section we show how to add a new clusterhead to the NTDR backbone in such a way that it becomes part of the DCA. This operation could either take place during the exchange of the signaling messages when the new entity joins the backbone, or take place independently, soon after the backbone join has been accomplished. Either way, the new clusterhead requires a share of the existing DCA private key.

After bootstrapping, assume that there are n clusterheads who hold DCA private key shares. Each of these clusterheads retains knowledge of one of the n secret polynomials that were used during bootstrapping. We will label any clusterhead with knowledge of one of these polynomials as an *active* clusterhead and refer to n as the *active threshold*. The DCA clusterhead join protocol requires that there are n active clusterheads and runs as follows for a new clusterhead CH_l:

1. Each active clusterhead creates a new subshare in the same way as during Step $3)$ of the bootstrapping. In other words, CH_i generates a subshare $f_i(l)$. This subshare is then securely unicast from CH_i to CH_l (if CH_l is new to the network then this is done by by encrypting it with the public key corresponding to the offline certificate held by CH_l).
2. CH_k computes their share of the DCA private key as $x_l = \sum_{i=1}^{n} f_i(l)$.

Note that after this protocol has been run, CH_l possesses a share x_l of SK but does not have knowledge of any of the bootstrap secret polynomials. We refer to clusterheads of this type as *passive* clusterheads, since they can assist in providing DCA services but cannot assist in new DCA clusterhead joins.

It is important to maintain n active clusterheads at all times, in order to support new clusterhead joins. If one of the active clusterheads is lost then a passive clusterhead must immediately be "promoted" to replace the departing clusterhead. If active clusterhead CH_i leaves then passive clusterhead CH_l can be promoted by any k remaining active clusterheads as follows:

1. Each of the k active clusterheads CH_j securely unicasts $f_i(j)$ to CH_l. (This is a value that was unicast from CH_i to CH_j during the bootstrap protocol.)
2. CH_k reconstructs f_i by polynomial interpolation.

Note that the timely information that an active clusterhead has left, and its identity, are both easily obtained from the regular beacon messages that are exchanged between clusterheads. This allows the above protocol to ensure that we always have n active clusterheads in our DCA.

3.9 DCA Clusterhead eviction

A clusterhead eviction could happen as a result of normal operational reasons such as instant unavailability or communication failure, or security reasons such as cheating, node compromise, or identification failure. These two cases have slightly different implications. If the reason for eviction is unclear then we should assume the latter case and consider the clusterhead compromised.

If a clusterhead is evicted for operational reasons and is not regarded as a security vulnerability then it suffices to check whether the evicted clusterhead is active or passive. If it is passive then no action is required. If it is active then the promotion protocol of Section 3.8 should be run to elect a new active clusterhead.

If a clusterhead is evicted for security reasons then the share of the DCA private key must be considered compromised. If the evicted clusterhead is active then we must assume likewise that the associated secret polynomial is compromised. There are several different options:

- One option is just to accept a degree of risk. Since k clusterheads are required to provide DCA services, the compromise of one clusterhead means at worst that it still requires $k-1$ remaining clusterheads to collude with the evicted clusterhead before the DCA private key can be obtained. In some environments this might be acceptable. It may be the case that it requires a certain number of evictions to take place before the system is no longer deemed operational.
- At the other end of the scale, if a clusterhead compromise is regarded as extremely sensitive then the whole DCA could be bootstrapped again. This not only has implications with respect to communication cost, but potentially means that all existing online certificates would require renewal under the new DCA key pair.
- It is possible to design a DCA in which DCA shareholders can be evicted by adopting a threshold scheme that offers *disenrollment* capabilities (for example [2], [5]). However, such schemes typically only allow a limited number of evictions and increase the storage requirements of each clusterhead.
- A more acceptable compromise is probably to run the share refresh protocol that we describe in Section 3.10. This involves a similar communication cost to bootstrapping but does not change the DCA private key.

In most of the above scenarios it is still necessary to first run the promotion protocol in the event that the evicted clusterhead is active.

3.10 Refreshing DCA clusterhead shares

It is good practice to periodically *refresh* the DCA private key shares. We note that it is desirable to perform this process more often than refreshing the private key (which essentially requires the system to be bootstrapped once again). The purpose of the share refresh protocol is thus to refresh the shares without changing the associated private key SK that they protect. This process is sometimes referred to as *proactive* secret sharing [9] and has been proposed as countermeasure to *mobile adversaries* [14], who attempt to compromise clusterheads over a period of time.

Our share refresh protocol is based on the bootstrap protocol and uses the underlying idea of [9]. Let n be the number of active clusterheads and m be the total number of clusterheads.

1. Each active clusterhead CH_i creates a (k, m) threshold sharing of the secret value 0 by generating a polynomial function $h_i(z) = \sum_{l=1}^{k-1} b_{i,l}x^l$ of degree at most $k-1$. Note that $h_i(0) = 0 \, mod \, p$.
2. CH_i uses a secure unicast channel to distributes the subshare $h_i(j)$ to clusterhead CH_j.
3. CH_i broadcasts the values $y_{i,l} = g^{b_{i,l}}, (l \in \{0, .., k-1\})$.
4. Each CH_j verifies that the subshare $h_i(j)$ received from CH_i is valid by checking that $g^{h_i(j)} = \prod_{l=0}^{k-1}(y_{i,l})^{j^l}$. If this equality holds then the value received from CH_i is correct. If not then CH_j issues a warning. If any CH_i has been the target of a warning then they are asked to repeat from Step 1).
5. Once there are no active warnings, each CH_j computes $x'_j = x_j + \sum_{i=1}^{n} h_i(j)$. Active clusterhead CH_i refreshes its secret polynomial to $f'_i(z) = f_i(z) + h_i(z)$.

At the end of the above protocol each active clusterhead CH_j holds a share x'_j of the DCA private key $SK' = SK + \sum_{j=1}^{n} h_j(0) = SK + 0 = SK$.

Note that the above share refresh protocol requires a similar communication cost to the bootstrap protocol on which it is based. One way of reducing this could be to simplify the protocol by using just a few of the active clusterheads (possibly just one). This would still result in all DCA clusterhead shares being refreshed, although not all secret polynomials would be refreshed. It also leaves the problem of which active clusterheads to trust with such a role in the share refresh protocol.

4 Alternative approaches

We now briefly review a number of alternative approaches to distributing CA services in an ad hoc network.

In [23] a threshold DCA was proposed. This system also uses a version of threshold DSS. In contrast to our shared key generation, this system is initialized off-line by a dealer who issues DCA private key shares to a collection of special server nodes. Protocols for dynamic node management to support this DCA were not proposed.

Another approach was taken in [11], where any node can play the role of server. In this approach a dealer (which is a mobile node) initializes k nodes with shares of a RSA-based private key, which these nodes then propagate through the network. If a node wants to sign a request by the DCA, a threshold of nodes must be in the vicinity (one routing hop). A join protocol was proposed that involves the cooperation of k existing nodes and two rounds of secure unicast exchange. This approach requires a dealing node to be entrusted with the DCA private key.

In [3] a DCA was proposed for cluster-based ad hoc networks which, like our scheme, uses clusterheads as DCA shareholders. However explicit clusterhead join protocols were not proposed. Moreover, in order to grant affiliation status to clusterheads, this scheme relies on a trust relationship based on warranty certificates issued by the nodes themselves. Hence, the realization of such a proposition is based on whether such a trust relationship amongst nodes exists.

An alternative to establishing a structured DCA in ad hoc networks is the *self-organized public infrastructure* proposed in [10]. This approach is similar to PGP [8] in the way that the nodes issue certificates to one other based on node relationships. Each node keeps a local certificate repository which contains certificates of other nodes selected in a defined way. The drawbacks of this approach are that the use of the certificate repository leads to some overhead and that the approach is based on transitive trust, which is not always scalable in the case of large ad hoc networks.

5 Conclusion

In this paper we have proposed a distributed certificate authority designed to provide public key services in an NTDR ad hoc network. We have outlined a number of supporting protocols, largely based on established techniques, for creating and running such services in a dynamic environment. While this system has been designed for operation within an NTDR security architecture, most of the techniques could be adapted for deployment in any cluster-based ad hoc network and so it is hoped that this system will be of wider interest.

REFERENCES

[1] B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens. An on-demand secure routing protocol resilent to byzantine failures. In *ACM Workshop on Wireless Security (WiSe 2002)*, September 2002.

[2] S.G. Barwick, W.-A. Jackson, and K.M. Martin. Updating the parameters of a threshold scheme by minimal broadcast. *IEEE Trans. Inf. Theory*, 51(2):620–633, 2005.

[3] M. Bechler, H. J. Hof, D. Kraft, F. Pahlke, and L. Wolf. A cluster-based security architecture for ad hoc networks. In *IEEE INFOCOM*, 2004.

[4] B. Blakley. Safeguarding cryptographic keys. In *Proceedings AFIPS 1979 National Computer Conference*, pages 313–317, June 1979.

[5] B. Blakley, G.R. Blakley, A. Chan, and J. Massey. Threshold schemes with disenrollment. In *Adv. in Cryptology - CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 540–548. Springer-Verlag, 1993.

[6] S. Buchegger and J.Y. Le Boudec. Cooperative routing in mobile ad-hoc networks: Current efforts against malice and selfishness. In *Mobile Internet Workshop, Informatik 2002*, Lecture Notes on Informatics. Springer-Verlag, October 2002.

[7] B. Dahill, B. N. Levine, E. Royer, and C. Shields. A secure routing protocol for ad hoc networks. In *Proceedings of the Tenth Conference on Network Protocols (ICNP)*, November 2002.

[8] S. Garfinkel. *PGP: Pretty Good Privacy*. O'Reilly & Associates, 1995.

[9] Amir Herzberg, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive secret sharing or: How to cope with perpetual leakage. *Lecture Notes in Computer Science*, 963:339–352, 1995.

[10] J. Hubaux, L. Buttyan, and S. Capkun. The quest for security in mobile ad hoc networks.

[11] Jiejun Kong, Petros Zerfos, Haiyun Luo, Songwu Lu, and Lixia Zhang. Providing Robust and Ubiquitous Security Support for Wireless Mobile Networks. In *Ninth International Conference on Network Protocols (ICNP'01)*, pages 251–260, 2001.

[12] S. Lee, B. Han, and M. Shin. Robust routing in wireless ad hoc networks. In *2002 International Conference on Parallel Processing Workshops (ICPPW'02)*, August 2002.

[13] P. Michiardi and R. Molva. Ad hoc network security, 2003. P. Michiardi and R. Molva, Ad hoc network security, ST Journal of System Research, Volume 4, N1, March 2003.

[14] R. Ostrovsky and M. Yung. How to withstand mobile virus attacks. In *Proceedings of the 10th ACM Symposium on principles of Distributed Computing*, pages 51–59, 1991.

[15] P. Papadimitratos and Z. J. Haas. Secure link state routing for mobile ad hoc networks. In *IEEE Workshop on Security and Assurance in Ad hoc Networks, Orlando, FL*, January 2003.

[16] C. Park and K. Kurosawa. New elgamal type threshold digital signature scheme. *IEICE Trans. Fundamentals*, E79-A(1):86–93, 1996.

[17] T. P. Pederson. A threshold cryptosystem without a trusted party. In *Eurocrypt '91*, volume 547 of *Lecture Notes in Computer Science*, pages 522–526. Springer-Verlag, 1991.

[18] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[19] V. Varadharajan. Security for cluster based ad-hoc networks, July 2002. Tech Report, Loria.

[20] K. Vesa. Security in ad hoc networks, 2000. Karpijoki, Vesa. Security in Ad hoc Networks, In Proceedings of the Helsinki University of Technology, Seminars on Network Security, Helsinki, Finland, 2000.

[21] G. M. Zapata and N. Asokan. Securing ad-hoc routing protocols. In *ACM Workshop on Wireless Security (WiSe 2002)*, September 2002.

[22] J. Zavgren. NTDR Mobility Management Protocols and Procedures, November 1997. In Proceedings of the IEEE Military Communications Conderence (MILCOM'97).

[23] Lidong Zhou and Zygmunt J. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24–30, 1999.