Proceedings of the 2004 IEEE
Workshop on Information Assurance and Security
United States Military Academy, West Point, NY, 10–11 May 2004

Secure Communications in Ad hoc Networks

Mike Burmester and Tri Van Le
Department of Computer Science, Florida State University, Florida 32306-4530,
{*burmester, levan*}@cs.fsu.edu, Tel: (850) 644-6410, Fax: (850) 644-0058.

Abstract— **Ad hoc networks are collections of mobile nodes with links that are made or broken in an arbitrary way. They have no fixed infrastructure and may have constrained resources. The next generation of IT applications is expected to rely heavily on such networks. However, before they can be successfully deployed several major security threats must be addressed. These threats are due mainly to the ad hoc nature of these networks. Consequently it may be much harder (or even impossible) to establish a secure communication channel that can tolerate malicious faults. In this paper we first propose a general model for ad hoc networks based on Bayesian inferences that satisfies the basic mobility requirements of such networks and formally define our requirements for secure communication. We then propose a secure communication protocol that will trace malicious faults.**

I. Introduction

Ad hoc networks are dynamic collections of self-organizing mobile nodes with links that are changing in an unpredictable way. They are characterized by a dynamic topology and the lack of any fixed infrastructure. The communication medium is broadcast. The nodes can be regarded as wireless mobile hosts with limited power, constrained bandwidth and transmission range. The recent rise in popularity of mobile wireless devices and technological developments has made possible the deployment of such networks for several applications. Indeed, because ad hoc networks do not have any fixed infrastructure such as stations or routers, they are highly applicable for emergency deployments, disasters, search and rescue missions and military operations.

Finding and maintaining routes in an ad hoc network is a major challenge. So far, most of the research has focused on functionality issues and efficiency (e.g., [3, 6, 23, 24, 27, 31, 32, 35]) with security being given a lower priority, and in many cases, regarded as an add-on afterthought technology rather than a design feature (e.g., [2, 25, 33]). Although such an approach may be suitable for networks with predictable faults, it not suitable for ad hoc networks in which we have unpredictable or malicious. Of particular concern is the possibility that an established route is under the control of a malicious adversary, and will be disconnected at a critical time when damage is maximized.

In this paper we describe a general model for ad hoc networks that satisfies the basic mobility requirements of such networks and formally define our security requirements. We then present a communication algorithm that is secure and will trace malicious faults. Security is proven in our formal model.

This material is based on work supported in part by the U.S. Army Research Laboratory and the U.S. Research Office under grant number DAAD19-02-1-0235.

II. A model for ad hoc networks

There are several ways in which one can model the unpredictable nature of an ad hoc network. Whichever way is used, there are important mobility aspects that must be reflected in the model. Clearly the model has to be time dependent and stochastic. That is, there must be a probability distribution $p_{ij}^t = \Pr[(x_i, x_j) = 1]$ defined on all possible links (x_i, y_j) of the network. Here p_{ij}^t is the probability that pair of distinct nodes (x_i, y_j) is linked at time t. Since the ad hoc-ness of the system is due to the mobility of the nodes, the model should reflect this. In particular, the link probability distribution should have *memory*. Consequently we model an ad hoc network by using a *Bayesian inference* structure for which established links have a high probability to remain so and a low probability to be disconnected. Similarly for disconnected links. –see Figure 1. That is, the à posteriori probability that x_i, x_j

$$
\begin{aligned}
\Pr[(x_i,x_j)^t = 0 \mid (x_i,x_j)^{t-1} = 0] &= a_{ij}^{0t} \quad (high),\\
\Pr[(x_i,x_j)^t = 1 \mid (x_i,x_j)^{t-1} = 0] &= 1 - a_{ij}^{0t}\ (low),\\
\Pr[(x_i,x_j)^t = 1 \mid (x_i,x_j)^{t-1} = 1] &= a_{ij}^{1t} \quad (high),\\
\Pr[(x_i,x_j)^t = 0 \mid (x_i,x_j)^{t-1} = 1] &= 1 - a_{ij}^{1t}\ (low),
\end{aligned}
$$

Fig. 1. The basic inference requirements for an ad hoc network

are linked given that x_i, x_j where previously linked is high, and similarly for the non-linked case. The link probabilities are determined jointly by the nodes, Nature and possibly the adversary. The contribution of the nodes comes from their mobility. Nature's contribution comes from the fact that communication is wireless. A wide range of environmental factors may affect communication, ranging from weather and radio interference to physical obstacles. Finally there are scenarios in which the adversary influences the mobility of the system, as for example in "seek and destroy" missions. These probabilities may also be linked by Markov interdependencies, to reflect the particular nature of the node mobility. The Bayesian model supports the automatic derivation of probabilities for a set of possible causes and supports a stochastic infrastructure.

Definition 1: A Turing Machine [1] equipped with an input tape, an output tape, i read-only communication tapes, j write-only communication tapes, a work tape and a random tape, is called an (i,j)-*Interactive Turing Machine* ((i,j)-ITM). We only consider *polynomially bounded* ITM's in this paper.

Definition 2: Let $\mathcal{G} = \{(V, L^t)\}$ be a family of networks,[1] with node set V, $m = |V|$, and link sets L^t indexed by $t \in Z^+$, where t is time, and let $\mathcal{N}$ be Nature, such that:

1. Each node $x_i \in V$ is a $(m-1,m)$-ITM and $\mathcal{N}$ is a $(m,0)$-ITM.
2. Nature $\mathcal{N}$ shares with each $x_i \in V$ a communication tape which is read-only for $\mathcal{N}$ and write-only for each x_i. Each pair of distinct nodes x_i, x_j share a communication tape which is read-only for x_i and write-only for x_j, and a communication tape which is write-only for x_i and read-only for x_j. The tapes are used in a specific way and are subject to the following constraints:

(a) *Mobility assumption I.* Each node $x_i \in V$ writes at time t on its communication tape with $\mathcal{N}$ its mobility data, determined from data on its work tape at time $t-1$, randomness from its random tape and its input.

(b) *Mobility assumption II.* Nature $\mathcal{N}$ reads the mobility data of each $x_i \in V$ at time t from the communication tape it shares with x_i, and combines this with randomness from its random tape, data on its work tape at time $t-1$ and its input, to determine the transitional probabilities a_{ij}^{bt}, $b = 0,1$, and the link probabilities p_{ij}^t for all $x_j \in V$, $j \neq i$. $\mathcal{N}$ outputs all the values p_{ij}^t and a_{ij}^{bt}.

(c) *Initialization assumption.* The initial state (V, L^0) of the network is input to $\mathcal{N}$. This is used together with the transitional link probabilities to determine the links in L^t at time t.

(d) *Secret keys assumption.* If secret keys are used, these are input to the nodes x_i.

$\mathcal{G}$ is an *ad hoc* network, if the transitional link probabilities a_{ij}^{bt}, $b = 0,1$, computed by $\mathcal{N}$ in Step 2b satisfy the Bayesian inferences in Figure 1.

The Adversary $\mathcal{A}$ is a $(k,0)$-ITM who can corrupt up to k nodes $x_i' \in V$ throughout the lifetime of the system. $\mathcal{A}$ is called a k-adversary and the corrupted nodes, $f$aulty or $m$alicious[2] $\mathcal{A}$ shares with each of the faulty nodes x_i' a communication tape that is read-only for $\mathcal{A}$ and write-only for x_j, and a communication tape that is write-only for $\mathcal{A}$ and read-only for x_j. These tapes are used by $\mathcal{A}$ to co-ordinate the attacks of the network. $\mathcal{A}$ is a powerful adversary who has the accumulated knowledge of the k faulty nodes (whereas the knowledge of the non-faulty nodes is restricted to their local topologies.)

A. Routes and communication algorithms

Communication in $\mathcal{G}$ is achieved by forwarding packets via routes. A packet has two items: a header (with forwarding details) and a data item. A route $\mathcal{R}(s,d)$ is a path that links a *source* node s to a *destination* node d. That is, $\mathcal{R}(s,d) = (s = x_0, x_1, \ldots, x_r = d)$, with nodes in V and links in L^t.

Definition 3: Let $\mathcal{G} = \{(V, L^t)\}$ be an ad hoc network and $\mathcal{R}(s,d)$ a route. A multi-party algorithm $\mathcal{C}$ in $\mathcal{G}$ is a *communication algorithm* if on input $(\mathcal{R}(s,d), data)$ to s: s will output *success* or $failure$ and d will output $(s', data')$ or λ (the empty string).

[1] For simplicity we assume that the transmission range of all mobile nodes is the same. If this is not the same then we have to use directed graphs.

[2] We can generalize this to allow for a more dynamic scenario in which the adversary is able to replace some captured nodes by an equal number of new nodes.

$\mathcal{C}$ is (*conditionally*) *secure* if, for every pair of distinct non-faulty nodes s, d in V, any route $\mathcal{R}(s,d)$ in $\mathcal{G}$, any item *data* and any k-adversary $\mathcal{A}$:
1. *Privacy*: for any item $data^*$ having the same length as *data*, the probability that $\mathcal{A}$ will be able to distinguish which one of *data*, $data^*$ was given as input to $\mathcal{C}$ is at most $\frac{1}{2} + \varepsilon$, where ε is negligible (in the length of the description of $\mathcal{G}$).
2. *Robustness*: if $\mathcal{C}$ is input $(\mathcal{R}(s,d), data)$ then, the probability that d will output $(s, data))$ and s will output *success*, is at least $1 - \varepsilon$, where ε is negligible.

B. Cryptographic mechanisms

For privacy, *data* is encrypted. For integrity, Message Authentication Codes (MACs) are used. For authenticity and integrity, *data* is digitally signed. These are keyed mechanisms. There are two types of cryptosystems: *symmetric* and *public key*. Symmetric cryptosystems require one (shared) *secret* key. Public key cryptosystems require two keys, a *public key* and a *secret key*. In our algorithm below we shall use the following notation:
• $\{data\}_{xy}$: the encryption of *data* with the shared of x, y.
• $\{data\}_x$: the encryption of *data* with the public key of x.
• $\{[data]_{sd}\}_y$: the encapsulation of *data* obtained by first taking its MAC with the shared key of s, d and then encrypting it with a public key of y.
• $\{[data]\}_{xy}$: the encapsulation of *data* obtained by first taking its MAC and then encrypting it, both with shared keys of x, y.
The computational cost of public key cryptosystems is relatively high for ad hoc network applications. It is therefore preferable to use symmetric key mechanisms for encryption and message authentication. There are however cases when one has to use public key cryptosystems. To reduce the computational complexity, one can use Elliptic Curve (EC) cryptosystems [34], or the NTRU [29] cryptosystem.

III. Routing algorithms

There are two basic types of routing algorithms for ad-hoc networks that depend on where most of the routing effort takes place: network-centric and source-centric. With network-centric routing the routing effort is distributed within the network and is regarded as a service provided to sending nodes. With source-centric routing the source node is responsible for discovering the topology of the network, find the routes to the destination and update any changes with minimal help from other nodes. Paths to the destination are constructed on-demand and updated according to the changes in the network with cooperation from other nodes in the network limited to basic services such as forwarding packets and answering with neighborhood information.

¿From a security point of view network-centric routing involves substantial cooperation between the network nodes and thus requires strong trust relationships among the nodes. Source-centric routing lessens these requirements and thus is less vulnerable to malicious attacks. In this paper we focus on source-centric routing algorithms and consider ways to support their security by combining appropriate path structures with cryptographic or other mechanisms. We start by analyzing the security threats.

A. Security issues

Ad hoc networks are vulnerable to all types of faults that occur in fixed infrastructure networks, but are particularly vulnerable to malicious faults. In many cases there is no way to prevent such faults because the routing service is provided by remote nodes that may be faulty. In particular, the source node cannot in general distinguish a malicious link-break from an ordinary link-break caused by Nature $\mathcal{N}$. Passive (eavesdropping) attacks can easily be controlled by using encryption mechanisms. Although integrity and authentication mechanisms may also be used to restrict active attacks, they may not be sufficient.

Man-in-the-middle attacks. In a man-in-the-middle attack or "tunnel" attack the adversary takes control of the communication channel between the source and destination by interposing between them [5]. In their simplest form, man-in-the-middle attacks are *passive relay attacks* in which packets between the source and destination are simply relayed via nodes under the control of the adversary. The attacker is transparent and the source is fooled into believing that the destination is much closer (in broadcast hops) than it actually is. Consequently a path under the control of the adversary may be selected in preference to other paths, that may be shorter. There are several variants of a man-in-the-middle attack. The hardest to control are active attacks that involve "insider" nodes, that is malicious nodes that are trusted. As in the passive case, the attacker "tunnels" packets intended for the destination via nodes under his control. Only this time, some of

the faulty nodes are insiders, listed on the route to the destination. It is much harder to deal with insider attacks because in general it may not be possible to distinguish a non-faulty node from a malicious node in networks that do not have a fixed infrastructure. This is because the adversary may disguise an attack in such a way that it "mimics" (stochastically) a fault caused by Nature.

Active man-in-the-middle attacks come in several flavors including *Rushing attacks* [21] and *Sybil attacks* [15]. In a Rushing attack the adversary succeeds in fooling the source into believing that a route is short, by relaying packets much faster through nodes under his control. In a Sybil attack a malicious node x presents multiple identities x_i. In this way x succeeds in fooling the source into believing that there are many short routes to the destination, all through malicious nodes x_i that may actually be far away (in broadcast hops), but which are used as a proxy by the "nearby"node x.

Tracing malicious nodes. Link-breaks caused by Nature in ad hoc networks are fundamentally different from malicious link-breaks. This is because Nature's faults are usually independent, in the sense that they do not occur in a coordinated way, as is generally the case with Byzantine faults. In particular Nature's actions are independent of those of the adversary and affect both the adversary and the (non-corrupted) network nodes in the same way.[3] In our general model it is not possible to trace Nature's faults directly (when the network is not simulatable). However it is possible to trace "abnormal" faults by exploiting the fact that such behavior can be detected by non-faulty neighbor nodes.[4] Below we describe such an algorithm.

B. A Communication algorithm that traces malicious faults

The algorithm that we describe does not rely on a stochastic analysis and is not restricted to attacks that occur with a particular frequency. It attributes faults to neighbor pairs of nodes, and therefore may implicate with each malicious node a non-faulty node,[5] if no other information is available. Although this may be a high price to pay in some cases, in general it is reasonable if one takes into account the potential damage that a malicious node can cause. If several such pairs are traced by this algorithm for different source/destination pairs, then it may be possible to single-out individual malicious nodes.

Our algorithm uses a *timer* to trace malicious nodes. Let $\mathcal{R}(s,d) = (s{=}x_0, x_1, \ldots, x_n{=}d)$ be the route used and *data* the message that the source s wants to send to the destination d. When there are no faults, packets that contain the encapsulation of *data* are forwarded to d via the intermediate nodes on $\mathcal{R}$. If the packet received by d is valid, then d responds with an acknowledgement ack_d. If there is a fault, caused say by the malicious node x_{i+1}, then node x_i will not receive in due course the acknowledgement ack_d and will send a negative acknowledgement $nack_i$ downstream to s. The source s uses this report to trace malicious faults.

We use shared keys for encryption and authentication except when malicious faults occur. If privacy is not require then there is no need to encrypt the packet.

We now describe the algorithm in detail. Let id_x be an identifier for x, *counter* a counter that is incremented with each successful transmission, t_x a timer used by node x and τ an upper time bound for a one-hop round-trip. In the following, denote $pkt_s = [id_s, id_d, counter, \{data\}_{sd}]_s$ the data packet sent by s; $ack_d = [id_s, id_d, counter]_d$ and $nack_i = [id_s, id_d, id_{x_i}]_{x_i}$ are the acknowledgements.

SecureCom $(\mathcal{R}(s,d) = (s = x_0, \ldots, x_r = d); data)$

Source: $s(\mathcal{R}(s,d), data)$.

1. $s \to x_1 : pkt_s$.
2. Start timer $t_s = n\tau$.
3. While timer t_s has not expired:
   (a) If a valid ack_d is received then
      i. set $counter = counter + 1$
      ii. stop timer t_s and return $success$.
   (b) If a valid $nack_i$ is received then
      i. stop timer t_s and return $failed(x_i, x_{i+1})$.
4. If timer t_s expires then return $failed(x_0, x_1)$.

Intermediate node x_i $(0 < i < n)$.
If a valid pkt_s is received then do the following:

1. $x_i \to x_{i+1} : pkt_s$.
2. start timer $t_i = (n-i)\tau$.
3. While timer t_i has not expired:

[3] In the extreme ad hoc case when links are made or broken with probability 0.5, the adversary is for all practical purposes impotent.

[4] Malicious attacks that "mimic" Nature's mobility faults cannot be traced but can be controlled by using appropriate redundancy.

[5] Awerbuch et al. [2] use an algorithm that has two distinct phases: a communication and a probing phase. The latter is triggered if "abnormal" behaviour is detected. Such an approach may fail because a "cunning" adversary will recognize when probing has started and behave faultlessly during probing.

(a) If a valid ack_d is received then
  i. $x_i \rightarrow x_{i-1} : ack_d$.
  ii. stop timer t_i and terminate.
(b) If a valid $nack_j$, $i<j<n$, is received then
  i. $x_i \rightarrow x_{i-1} : nack_j$.
  ii. stop timer t_i and terminate.

4. If timer t_i expires then
(a) $x_i \rightarrow x_{i-1} : nack_i$.

Destination: d.
If a valid pkt_s is received then do the following:
1. $ack_d \rightarrow x_{n-1}$ and return $data$.

end of algorithm

In this algorithm: *success* means the packet has been delivered to d successfully; $failed(x_i, x_{i+1})$ means that either x_i or x_{i+1} is malicious or the link (x_i, x_{i+1}) is broken.

$$s \xrightarrow{pkt_s} x_1 \xrightarrow{pkt_s} \ldots \xrightarrow{pkt_s} x_i \xrightarrow{pkt_s} x_{i+1} \xrightarrow{pkt_s} \ldots \xrightarrow{pkt_s} d$$
$$s \xleftarrow{ack_d} x_1 \xleftarrow{ack_d} \ldots \xleftarrow{ack_d} x_i \xleftarrow{ack_d} x_{i+1} \xleftarrow{ack_d} \ldots \xleftarrow{ack_d} d$$

Fig. 2. A flow diagram for $SecureCom$: no faults

$$s \xrightarrow{pkt_s} x_1 \xrightarrow{pkt_s} \ldots \xrightarrow{pkt_s} x_i \xrightarrow{pkt_s} x_{i+1}$$
$$s \xleftarrow{nack_i} x_1 \xleftarrow{nack_i} \ldots \xleftarrow{nack_i} x_i \leftarrow\!\!- x_{i+1}$$

Fig. 3. A flow diagram in which x_{i+1} does not forward pkt_s

$$s \xrightarrow{pkt_s} x_1 \xrightarrow{pkt_s} \ldots \xrightarrow{pkt_s} x_i \xrightarrow{pkt_s} x_{i+1} \xrightarrow{pkt'_s} x_{i+2}$$
$$s \xleftarrow{nack_i} x_1 \xleftarrow{nack_i} \ldots \xleftarrow{nack_i} x_i \leftarrow\!\!- x_{i+1} \qquad x_{i+2}$$

Fig. 4. A flow diagram in which x_{i+1} substitutes pkt_s by pkt'_s

In the last two diagrams, the dashed arrow originating from x_{i+1} means x_{i+1} may optionally send a $nack_{i+1}$ to x_i, in which case, s and other nodes on the left of x_i receive $nack_{i+1}$ instead of $nack_i$.

Theorem 1: Let $\mathcal{R}(s,d)$ be a route that is not subject to ad hoc faults. Suppose that $(data, \mathcal{R}(s,d))$ is input to $SecureCom$ and that a semantically secure encryption scheme and a digital signature scheme that is secure cryptographic mechanisms are used [34]. Then $SecureCom$ is either:

- a secure communication algorithm, or
- it will trace an adjacent pair of nodes (x_i, x_{i+1}) on $\mathcal{R}(s,d)$ such that at least one of x_i, x_{i+1} is malicious.

Proof. We only consider the case when s, d are not malicious. First suppose that the adversary is passive. Then it is easy to see that $SecureCom$ is a secure communication algorithm (Definition 3). Indeed, we get privacy because encryption semantically secure. Robustness follows because the adversary is passive.

Next suppose that the adversary is active. We shall prove that $SecureCom$ will trace an adjacent pair of nodes of which at least one is malicious. Observe that when there are no ad hoc faults:
1. Every node x_i, $0 < i < n$, will sent back either an *ack* or a *nack* to x_{i-1} by the time timer t_i expires.
2. If s receives a valid ack_d within time t_s then there are no faults.
3. If s receives a valid $nack_i$ within time t_s then either x_i or x_{i+1} is malicious, or both, since only x_i can compute $nack_i$.
4. If s (or x_i) does not receive a valid ack_d from x_1 (or x_{i+1}) within time t_s then x_1 (or x_{i+1}) is malicious.

Suppose that x_{i+1} is the first upstream malicious node on $\mathcal{R}(s,d)$. If x_{i+1} does not forward pkt_s or corrupts it so that d will not received a valid pkt_s, then, eventually, x_i will send downstream a $nack_i$ to s (Observation 4). Similarly, if x_{i+1} does not send downstream an ack_d or a $nack_j$ $(j > i+1)$ then x_i will eventually send $nack_i$ to s. Observe that $SecureCom$ is essentially a one pass protocol in which packets are encapsulated using semantically secure encryption and digital signatures that are secure against adaptive chosen message attacks. Therefore its security extends to Random Oracle model [4]. □

Note that when $failed(x_i, x_{i+1})$ is returned and an ad hoc fault (caused by Nature) has been excluded then it is *impossible* to tell which one of x_i and x_{i+1} is malicious, or if both are malicious, since we cannot decide which one lies.

Remark 1: The usefulness of this protocol derives from the fact that when there are no faults, a small *ack* is sent back. When faults do occur, a small *nack* packet is sent back. In both cases, in one round, a packet is confirmed successfully delivered, or a fault location is determined by using only one digital signature. This protocol greatly improves on the fault tracing algorithm in [2], where at least $\log(n)$ communication rounds are needed to locate a Byzantine fault.

Both *acks* and *nacks* must be verified. Therefore

one should use signature schemes for which verification is fast (e.g., RSA with $e = 3$). In the journal version of this paper we will show how the algorithm can be extended to cover the case where only symmetric key mechanisms are used.

IV. Extensions for authenticated key exchange

Our protocol is a one pass protocol in which the source and destination s, d share a secret key. For most applications shared keys must be refreshed, to prevent known plaintext/ciphertext attacks and for forward secrecy [4]. However there are applications in which this may not be possible, e.g., with sensor networks or when there are power limitations.

Several protocols can be used for authenticated (session) key exchange which are provably secure. Depending on the security model, we may either use the Needham-Shroeder protocol [28], with security proven in the BAN model or the strand space model [11, 17], or the AKE protocol by Bellare and Rogaway [4]. This last protocol is proven secure in the Random Oracle model [4]. Authenticated key exchange protocols which are provably secure are 3 pass protocols in which an entity s and an entity d in turn send each other data that enables them to exchange an authenticated session key (see e.g., [4, 28]).

Pass 1. $s \longrightarrow \ \ldots \ \longrightarrow d$
Pass 2. $d \longrightarrow \ \ldots \ \longrightarrow s$
Pass 3. $s \longrightarrow \ \ldots \ \longrightarrow d$

In the last pass s, d have exchanged sufficient data to be able compute the session key.

SecureCom is essentially a one pass protocol (when the adversary is passive) with security proven in the Random Oracle model. Therefore when combined with the AKE protocol (in the 3^{rd} pass), will give us a communication algorithm that is proven secure in the Random Oracle model. To allow for malicious attacks in the first two passes, we have to extend the tracebility subroutine of *SecureCom* to these passes (data must be encapsulated and *acks*, *nacks* must be used appropriately).

V. Extension For Reactive Tracing

We now show how malicious faults can be traced without signature verifications all the time, by using at most one more pass which is reactive only when faults occured. For tracing purpose, we require that each intermediate node stores all data packets it receives during preceeding two round-trip time period. When faults occur, each intermediate node starts verifying its received packets for packet drops and corruptiongs, as done in *SecureCom*, using logged data.

We use the same notations as before. Additionally, let $trace_s = [id_s, id_d, counter, hash(pkt_s)]_s$; and let $hash(x)$ be a collision-resistant hash function.

*SecureCom*2 ($\mathcal{R}(s,d) = (s = x_0, \ldots, x_r = d); data$)

Source: $s(\mathcal{R}(s,d), data)$.
1. $s \rightarrow x_1 : pkt_s$.
2. If a valid ack_d is received within time $n\tau$ then
   (a) set $counter = counter + 1$
   (b) return *success*.
3. $s \rightarrow x_1 : trace_s$.
4. If a valid $nack_i$ is received within time $n\tau$, then return $failed(x_i, x_{i+1})$, else return $failed(x_0, x_1)$.

Intermediate node x_i $(0 < i < n)$.
1. If a packet pkt_s is received then:
   (a) $x_i \rightarrow x_{i+1} : pkt_s$
   (b) store pkt_s in packet cache for $2n\tau$ time.
2. If a packet ack_d is received then:
   (a) $x_i \rightarrow x_{i-1} : ack_d$.
3. If a valid packet $trace_s$ is received then:
If there is a matching pkt_s in the cache then:
   (a) $x_i \rightarrow x_{i+1} : trace_s$.
   (b) If a valid $nack_j$ $(i < j < n)$ is received within time $(n-i)\tau$, then $x_i \rightarrow x_{i-1} : nack_j$, else $x_i \rightarrow x_{i-1} : nack_i$.

Destination: d.
If a valid pkt_s is received then do the following:
1. $ack_d \rightarrow x_{n-1}$ and return *data*.

end of algorithm

Remark 2: It is easy to see that the flow of *SecureCom*2 starting with $trace_s$ is the same as the flow of *SecureCom* starting with pkt_s, where pkt_s is replaced by $trace_s$. Thus Theorem 1 is proven for *SecureCom*2 identically. Furthermore, when there are no faults, *SecureCom*2 runs in two passes as *SecureCom*, but without any signature verification. When faults present, one signature signing and n signature verifications is expected.

Remark 3: For small footprint applications, each intermediate node x_i may choose to store $hash(pkt_s)$ instead of pkt_s in its packet cache.

In the above two diagrams, $\overset{*}{\leftarrow}$ means x_{i+1} may optionally send $nack_j$ to x_i, in which case, $nack_i$ is replaced by $nack_j$ in the diagrams. $\overset{*}{\dashrightarrow}$ means x_{i+1} may optionally send anything to x_{i+2}.

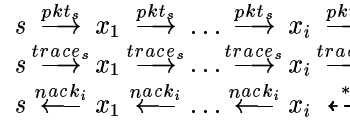

Fig. 5. A flow diagram in which x_{i+1} does not forward pkt_s

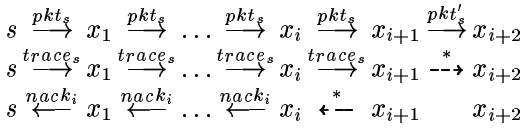

Fig. 6. A flow diagram in which x_{i+1} substitutes pkt_s by pkt'_s

After the first pass, there may be an optional pass where x_{i+1} constructs an ack_d. However, s will detect this and the tracing starts normally in the next pass.

VI. Conclusion

In this paper, we have presented a general model for ad hoc networks and formally defined the necessary requirements for secure communication in the presence of a Byzantine adversary. We then presented a secure communication protocol that traces malicious behaviors. The protocol can be adapted in such a way that it complements the security requirement of other routing protocols.

VII. Acknowledgements

The authors would like to thank an anonymous referee for requesting a more detailed analysis of the security model used (SectionIV).

References

[1] A. Aho, J. Hopcroft and J. Ullman, *The Design and Analysis of Computer Algorithms*, Reading MA, Addison Wesley, 1974.

[2] B. Awerbuch, D. Holmer, C. Nita-Rotaru and H. Rubens, *An On-Demand Secure Routing Protocol Resilient to Byzantine Failures*, ACM Workshop on Wireless Security (WiSe'02), 2002

[3] E.M. Belding-Royer and C.-K. Toh, *A review of current routing protocols for ad-hoc mobile wireless networks*, IEEE Personal Communications Magazine, pages 46-55, 1999

[4] M. Bellare and P. Rogaway. *Entity authentication and key distribution*. Advances in Cryptology - Crypto 93 Proceedings, Lecture Notes in Computer Science Vol. 773, D. Stinson ed, Springer-Verlag, 1994.

[5] S. Bengio, G. Brassard, Y. Desmedt, G. Goutier and J.J. Quisquater, *Secure implementations of identification systems*, Journal of Cryptology **4**(3), pp. 175–183, 1991

[6] J. Broch et al, *A performance comparison of multi-hop wireless ad hoc network routing protocols*, Proc. ACM MOBICOM, pp. 85-97, 1998.

[7] M. Burmester and Y. G. Desmedt, *Secure Communication in an Unknown Network Using Certificates*, Advances in Cryptology - Asiacrypt '99, LNCS # 1716, Springer, Berlin, pp. 274–287, 1999

[8] M. Burmester and Y. Desmedt and Y. Wang. *A critical analysis of models for fault-tolerant and secure computation.* Proceedings, Computer, Network and Information Security 2003, New York, 2003

[9] M. Burmester and Tri van Le. *Tracing Byzantine faults in ad hoc networks*, Proceedings, Computer, Network and Information Security 2003, New York, 2003

[10] M. Burmester and Tri van Le. *Secure Multipath Communication in Mobile Ad hoc Networks*, submitted to: International Conference on Information Technology, Coding and Computing, Las Vegas Nevada, April 5-7, 2004

[11] M. Burrows, M. Abadi, R. Needham and W. Stallings. *A Logic for Authentication*, in Practical Cryptography for Data Internetworks, IEEE Computer Society Press, 1996.

[12] S. Capkun, M. Hambdi and J. Hubaux, *Gps-free positioning in mobile ad hoc networks*, Proceedings of Hawaii Int. Conf. on System Sciences, 2001

[13] C.C. Chiang et al, *Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel*, Proceedings of IEEE SICON '97, pp. 197–211, April 1997.

[14] C. R. Davis, *IPSec: Securing VPNs*, McGraw-Hill, New York, 2000

[15] J. R. Douceur, *The Sybil attack*, Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), 2002

[16] R. Dube, C.D. Rais, K.Y. Wang, and S.K. Tripathi, *Signal Stability based Adaptive Routing for Ad-Hoc Mobile Networks*, IEEE Personal Communications, pp. 26–45, 1997

[17] F. Javier Thayer Fabrega and Jonathan C. Herzog and Joshua D. Guttman. *Strand Spaces: Proving Security Protocols Correct*, Journal of Computer Security, 7 (1999), pages 191-230.

[18] L.R. Ford and D.R. Fulkerson, *Flows in Networks. Princeton University Press*, Princeton, NJ, 1962.

[19] P. Georgatsos and D. Griffin, *A management system for load balancing through adaptive routing in multiserve ATM networks*, Proc. of IEEE Infocom, 1996.

[20] S. Goldwasser, S. Micali and C. Rackoff, *The knowledge complexity of interactive proof systems*, Siam J. Comp., **18**(1), pp. 186-208, 1989

[21] Y-C. Hu, A. Perrig and D.B. Johnson. *Rushing attacks and defense in wireless ad hoc network routing protocols*, WiSe 2003, pp. 30–40, 2003

[22] J.-P. HuBaux, L. Buttyan, and S. Capkun, *The quest for security in mobile ad hoc networks*, Proc. ACM Mobicom, 2001

[23] D.B. Johnson and D.A. Maltz, *Dynamic Source Routing in Ad-Hoc Wireless Networks*, Mobile Computing, ed. T. Imielinski and H. Korth, Kluwer Academic Publisher, pp. 152–181, 1996

[24] Y.B. Ko and N.H. Vaidya, *Location-Aided Routing in Mobile Ad Hoc Networks*, Proceedings of ACM/IEEE MOBICOM '98, 1998

[25] J. Kong et al., *Providing robust and ubiquitous security support for mobile ad-hoc networks*, Proc. IEEE ICNP, pp. 251-260, 2001

[26] N.F. Maxemchuk, *Dispersity routing in high-speed networks*, Computer networks and ISDN system 25, pp.645-661, 1993.

[27] S. Murphy and J.J. Garcia-Lunca-Aceves, *An efficient routing protocol for wireless networks*, ACM Mobile Networks and Applications Journal, pp. 182-197, 1996

[28] R. Needham and M. Schroeder, *Using Encryption for Au-*

thentication in Large Networks of Computers, Communications of the ACM, 21, pp.393-399, 1978.
[29] http://www.ntru.com/products/toolkits.htm
[30] V. Park and M. Corson, *A highly adaptive distributed routing algorithm for mobile wireless networks*, Proc. INFOCOM, April 1997.
[31] C.E. Perkins and P.Bhagwat, *Highly Dynamic Destination-Sequenced Distance-Vector Routing for Mobile Computers*, Computer Communications Review, pp. 224–244, 1994
[32] C.E. Perkins and E.M. Royer, *Ad hoc on-demand distance vector routing*, IEEE Workshop on Mobile Computing Systems and Applications, pp. 90-100, 1999
[33] K. Sanzgiri et al, *A Secure Routing Protocol for Ad Hoc Networks*,
citeseer.nj.nec.com/sanzgiri02secure.html
[34] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, John Wiley & Sons, Inc, New York, 1996
[35] S. Signh, M. Woo, and C.S. Raghavendra, *Power-Aware Routing in Mobile Ad Hoc Networks*, Proc. ACM/IEEE MOBICOM '98, October 1998
[36] H. Suzuki and F.A. Tobagi, *Fast bandwidth reservation scheme with multi-link and multi-path routing in ATM networks*, Proc. IEEE INFOCOM, 1992.
[37] C.K. Toh, *A Novel Distributed Routing Protocol To Support Ad-Hoc Mobile Computing*, Proceedings of 1996 IEEE Fifteenth Annual International Phoenix Conference on Computers and Communications, pp. 480–486, March 1996.
[38] S. Yi, P. Naldurg, and R. Kravets, *Security-aware ad hoc routing for wireless networks*, Proc. ACM Mobihoc, 2001
[39] L. Zhou and Z. J. Haas, *Securing ad hoc networks*, IEEE Network, 12(6):24-20, 1999