Provable Security of On-Demand Distance Vector Routing in Wireless Ad Hoc Networks

Gergely Ács, Levente Buttyán, and István Vajda

Laboratory of Cryptography and Systems Security (CrySyS),
Department of Telecommunications,
Budapest University of Technology and Economics, Hungary
{acs, buttyan, vajda}@crysys.hu

Abstract. In this paper, we propose a framework for the security analysis of on-demand, distance vector routing protocols for ad hoc networks, such as AODV, SAODV, and ARAN. The proposed approach is an adaptation of the simulation paradigm that is used extensively for the analysis of cryptographic algorithms and protocols, and it provides a rigorous method for proving that a given routing protocol is secure. We demonstrate the approach by representing known and new attacks on SAODV in our framework, and by proving that ARAN is secure in our model.

1 Introduction

Routing is a fundamental networking function, which makes it an ideal starting point for attacks aiming at disabling the operation of an ad hoc network. Therefore, securing routing is of paramount importance. Several "secure" routing protocols for ad hoc networks have been proposed in the academic literature (see [7] for a good overview), but their security have been analyzed by informal means only. In [3] and in [1], we show that flaws in routing protocols can be very subtle (leading to very sophisticated attacks), and therefore, they are very difficult to discover by informal reasoning. In [1], we propose a systematic approach based on a mathematical framework, in which the security of on-demand source routing protocols (e.g., DSR [8], Ariadne [6], and SRP [11]) can be analyzed rigorously. In this paper, we extend that approach to on-demand, distance vector routing protocols (e.g., AODV [12], SAODV [14] and ARAN [13]).

We must emphasize that by secure routing we mean the security of the route discovery part of the routing protocol only. In other words, we are not concerned with the problem of misbehaving nodes that do not forward data packets either for selfish or for malicious reasons. There are many attacks that aim at paralyzing the entire network by denial of service (e.g., rushing attack) or subverting the neighbor discovery mechanism (e.g., wormhole attack). In our notion these are not against the route discovery process primarily, and thus, we are not concerned with them in the rest of the paper.

R. Molva, G. Tsudik, and D. Westhoff (Eds.): ESAS 2005, LNCS 3813, pp. 113–127, 2005.

Rather, we focus on the problem of how to maintain the "correctness" of the routing information stored in the routing tables of the honest nodes in the presence of an adversary. We will define precisely what we mean by the "correctness" of routing table entries later in this paper.

Our mathematical framework is based on the simulation paradigm that has been successfully used to analyze the security of various cryptographic algorithms and protocols (see parts V and VI of [9] for an overview). In this approach, one constructs a real-world model that describes the real operation of the system, and an ideal-world model that captures what the system wants to achieve in terms of security. Then, in order to prove the security of the system, one proves that the outputs of the two models are indistinguishable (statistically or computationally). In [1], we apply this approach to source routing protocols, where the output of the models are sets of routes returned by the routing protocol in route reply messages. In case of distance vector routing, however, no routes are returned explicitly in the route reply messages. Hence, the main novelty of this paper is that here, the output of the models is the state of the system, which is represented by the content of the routing tables of the honest nodes.

The rest of the paper is organized as follows. In Section 2, we introduce our mathematical framework, which includes a precise definition of a "correct" system state, and based on that, a definition of routing security. Then, in Section 3, we illustrate the concepts introduced in Section 2 by representing known and new attacks on SAODV in our framework. In Section 4, we demonstrate the usefulness of our approach by formally proving that ARAN is secure in our model. Finally, we report on some related work in Section 5, and conclude the paper in Section 6.

2 Model

2.1 Static Representation of the System

Network model: We model the ad hoc network as an undirected labelled graph $G(V, E)$, where V is the set of vertices and E is the set of edges. Each vertex represents a node, and there is an edge between two vertices if and only if there is a radio link between the corresponding nodes. We assume that the radio links are symmetric, and that is why the graph is undirected.

We assume that the nodes use authenticated identifiers (e.g., public keys, symmetric keys) during neighbor discovery and in the routing protocol. We denote the set of identifiers by L, and we label each vertex v of G with the identifiers used by the node corresponding to v. We assume that honest (non-corrupted) nodes use a single identifier that is unique in the network, whereas corrupted nodes may use multiple compromised identifiers (see attacker model below). We represent the assignment of identifiers to the nodes by a labelling function $\mathcal{L} : V \to 2^L$, which returns for each vertex v the set of labels assigned to v. As we mentioned above, if v corresponds to a non-corrupted node, then $\mathcal{L}(v)$ is a singleton, and $\mathcal{L}(v) \not\subseteq \mathcal{L}(v')$ holds for any other vertex v'.

We also assign cost values to the nodes and to the radio links that may be interpreted as (minimum) processing and transmission costs, respectively, and may be used to compute routing metrics. The assignment of cost values is represented by two functions $\mathcal{C}_{node} : V \to \mathbb{R}$ and $\mathcal{C}_{link} : E \to \mathbb{R}$. Quite naturally, $\mathcal{C}_{node}(v)$ will represent the cost assigned to the node that corresponds to vertex v, and $\mathcal{C}_{link}(e)$ will represent the cost assigned to the link that corresponds to edge e. In the following, we will omit the indices *node* and *link* of $\mathcal{C}$ when the type of the argument unambiguously determines which of the two functions is used in a given context. An example for a typical cost assignment is the following: $\mathcal{C}(v) = 1$ for all $v \in V$, and $\mathcal{C}(e) = 0$ for all $e \in E$, which leads to the widely used hop count metric, where the cost of a route is equal to the number of intermediate nodes on the route.

Adversary model: We assume that the adversary is not all powerful, but it launches its attacks from corrupted nodes that it controls and that have similar communication capabilities as regular nodes. We denote the vertices that correspond to corrupted nodes by V^*. In addition, we assume that the adversary compromised some identifiers, by which we mean that the adversary compromised the cryptographic keys that are used to authenticate those identifiers. We denote the set of compromised identifiers by L^*. We further assume that the adversary distributed all compromised identifiers to all corrupted nodes, and hence, we have $\mathcal{L}(v) = L^*$ for all $v \in V^*$. Using the notation introduced in [6], the adversary described above is an Active-y-x adversary, where $x = |V^*|$ and $y = |L^*|$. In addition, we assume that the adversary is static in the sense that it does not corrupt more nodes and compromise more identifiers during the operation of the system.

Since neighboring corrupted nodes can communicate with each other in an unrestricted manner (e.g., by sending encrypted messages), they can appear as a single node (under all the compromised identifiers) to the other nodes. Hence, without loss of generality, we assume that corrupted nodes are not neighbors in G; if they were, we could merge them into a single corrupted node that would inherit all the neighbors of the original nodes.

Configuration: A configuration is a five tuple $(G(V,E), V^*, \mathcal{L}, \mathcal{C}_{node}, \mathcal{C}_{link})$ that consists of the network graph, the set of corrupted nodes, the labelling function, and the cost functions.

2.2 System States and Correctness

The state of the system is represented by the routing tables of the non-corrupted nodes. We assume that an entry of the routing table of a given node v contains the following three fields: the identifier of the target node, the identifier of the next hop towards the target, and the cost value that represents the believed cost of the route to the given target via the given next hop. Without loss of generality, we assume that the routing metric is such that routes with lower cost values are preferred.

Consequently, the state of the system in our model will be represented by a set $Q \subset (V \setminus V^*) \times L \times L \times \mathbb{R}$ of quadruples such that for any $(v, \ell_{tar}, \ell_{nxt}, c)$ and $(v', \ell'_{tar}, \ell'_{nxt}, c')$ in Q, $v = v'$ and $\ell_{tar} = \ell'_{tar}$ and $\ell_{nxt} = \ell'_{nxt}$ implies $c = c'$. The quadruple $(v, \ell_{tar}, \ell_{nxt}, c)$ in Q represents an entry in v's routing table with target identifier ℓ_{tar}, next hop identifier ℓ_{nxt}, and believed route cost c. The ensemble of quadruples that have v as their first element represent the entire routing table of v, and the ensemble of all quadruples in Q represent the ensemble of the routing tables of the non-corrupted nodes (i.e., the state of the system). Note that we allow that a node's routing table contains multiple entries for the same target, but the next hops should be different.

We define a correct state as follows:

Definition 1 (Correct state). *A state Q is correct if for every $(v,\ell_{tar},\ell_{nxt},c) \in Q$, there exists a sequence $v_1, v_2, \ldots, v_p$ of vertices in V such that $(v_i, v_{i+1}) \in E$ for all $1 \le i < p$, and*

- $v_1 = v$,
- $\ell_{tar} \in \mathcal{L}(v_p)$,
- $\ell_{nxt} \in \mathcal{L}(v_2)$, *and*
- $\sum_{i=2}^{p-1} \mathcal{C}_{node}(v_i) + \sum_{i=1}^{p-1} \mathcal{C}_{link}(v_i, v_{i+1}) \le c$.

Intuitively, the system is in a correct state if all the routing table entries of the non-corrupted nodes are correct in the sense that if v has an entry for target ℓ_{tar} with next hop ℓ_{nxt} and cost c, then indeed there exists a route in the network that

- starts from node v
- ends at a node that uses the identifier ℓ_{tar}
- passes through a neighbor of v that uses identifier ℓ_{nxt}, and
- has a cost that is smaller than or equal to c.

The requirement on the believed cost of the route (last point above) deserves some explanation. First of all, recall the assumption that routes with a lower cost are preferred. It is, therefore, natural to assume that the adversary wants to make routes appearing less costly than they are. This means that if node v believes that there exists a route between itself and target ℓ_{tar} (passing through neighbor ℓ_{nxt}) with a cost c, while in reality, there exist only routes between them with a cost higher than c, then the system should certainly be considered to be in an incorrect state (i.e., under attack). On the other hand, allowing the existence of routes with a smaller cost does not have any harm (under the assumption that the adversary has no incentive to increase the believed costs corresponding to the routes), and it makes the definition of the correct state less demanding. This has a particular importance in case of protocols that use one-way hash chains to protect hop count values (e.g., SAODV and alike), since in those protocols, the adversary can always increase the hop count by hashing the current hash chain element further. However, this ability of the attacker should rather be viewed as a *tolerable imperfection* of the system than a flaw in those protocols.

2.3 Dynamic Representation of the System

The simulation paradigm: The main idea of the simulation paradigm is to define two models: a real-world model that represents the behavior of the real system and an ideal-world model that describes how the system should work ideally. In both models, there is an adversary, whose behavior is not constrained apart from requiring it to run in time polynomial in the security parameter (e.g., size of the cryptographic keys used by the cryptographic primitives). This allows us to consider *any* feasible attacks, and makes the approach very general. Although the adversary is not constrained, the construction of the ideal-world model ensures that all of its attacks are unsuccessful against the ideal-world system. In other words, the ideal-world system is secure by construction.

Once the models are defined, the goal is to prove that for any real-world adversary, there exist an ideal-world adversary that can achieve essentially the same effects in the ideal-world model as those achieved by the real-world adversary in the real-world model (i.e., the ideal-world adversary can simulate the real-world adversary). A successful proof means that no attacks can be successful in the real-world model (or more precisely attacks can be successful only with negligible probability), since otherwise, an attack would be successful in the ideal-world model too, which is impossible by definition.

Real-world model: The real-world model that corresponds to a configuration $conf = (G(V, E), V^*, \mathcal{L}, \mathcal{C}_{node}, \mathcal{C}_{link})$ and adversary $\mathcal{A}$ is denoted by $sys^{\mathsf{real}}_{conf,\mathcal{A}}$, and it is illustrated on the left hand side of Figure 1. $sys^{\mathsf{real}}_{conf,\mathcal{A}}$ consists of a set $\{M_1, \ldots, M_n, A_1, \ldots, A_m, H, C\}$ of interacting Turing machines, where the interaction is realized via common tapes. Each M_i represents a non-corrupted device that corresponds to a vertex in $V \setminus V^*$, and each A_j represents a corrupted vertex in V^*. H is an abstraction of higher-layer protocols run by the honest parties, and C models the radio links represented by the edges in E. All machines are probabilistic.

We describe the operation of the real-world model only briefly, since it is essentially the same as the operation of the model described in [1]. Each machine is initialized with some input data (e.g., identifiers of neighbors, cryptographic keys, etc.), which determines its initial state. In addition, the machines also receive some random input (the coin flips to be used during the operation). Once the machines have been initialized, the computation begins. The machines operate in a reactive manner, which means that they need to be activated in order to perform some computation. When a machine is activated, it reads the content of its input tapes, processes the received data, updates its internal state, writes some output on its output tapes, and goes back to sleep. The machines are activated in rounds by a hypothetic scheduler (not illustrated in Figure 1). The order of activation is not important, apart from the requirement that C must be activated at the end of the round.

Machine C is intended to model the broadcast nature of radio communications. Its task is to read the content of the output tape of each machine M_i and A_j and copy it on the input tapes of *all* the neighboring machines, where the neighbor relationship is determined by the configuration *conf*. Machine H models higher-layer protocols (i.e., protocols above the routing protocol) and the end-users of the non-corrupted devices. H can initiate a route discovery process at any machine M_i by placing a request on tape req_i. A response to this request is eventually returned via tape res_i. Machines M_i $(1 \leq i \leq n)$ represent the non-corrupted nodes, which belong to the vertices in $V \setminus V^*$. M_i communicates with the other protocol machines via its output tape out_i and its input tape in_i, and its operation is essentially defined by the routing algorithm.

Finally, machines A_j $(1 \leq j \leq m)$ represent the corrupted nodes, which belong to the vertices in V^*. Regarding its communication capabilities, A_j is identical to any machine M_i. However, A_j may not follow the routing protocol faithfully. In addition, A_j may send out-of-band requests to H by writing on ext_j by which it can instruct the honest parties to initiate route discovery processes. Here, we make the restriction that the adversary triggers a route discovery only between non-corrupted nodes. Moreover, we restrict each A_j to write on ext_j only once, at the very beginning of the computation (i.e., before receiving any messages from other machines). This essentially means that we assume that the adversary is *non-adaptive*; it cannot initiate new route discoveries as a function of previously observed messages. Note, however, that each A_j can write multiple requests on ext_j, which means that we allow several parallel runs of the routing protocol.

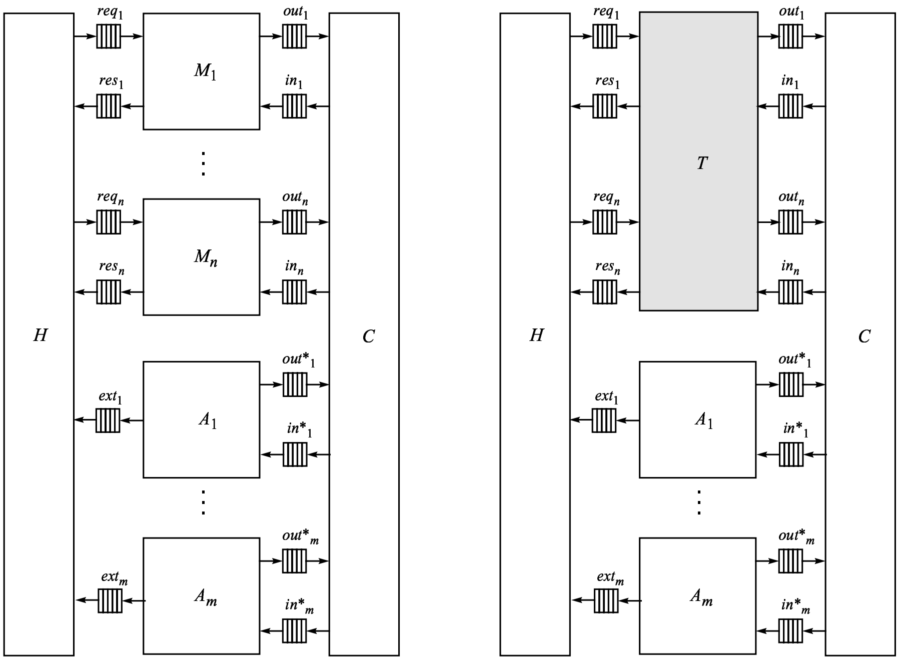

Fig. 1. The real-world model $sys^{\mathsf{real}}_{conf,\mathcal{A}}$ (left hand side) and the ideal-world model $sys^{\mathsf{ideal}}_{conf,\mathcal{A}}$ (right hand side)

The computation ends when H reaches one of its final states. This happens when H receives a response to each of the requests that it placed on the tapes req_i $(1 \leq i \leq n)$, where a response can also be a time-out. The output of $sys^{\mathsf{real}}_{conf,\mathcal{A}}$ is the ensemble of the routing tables of the non-corrupted nodes, which is a set of quadruples as defined above in Subsection 2.2. We denote the output by $Out^{\mathsf{real}}_{conf,\mathcal{A}}(r)$, where r is the random input of the model. In addition, $Out^{\mathsf{real}}_{conf,\mathcal{A}}$ will denote the random variable describing $Out^{\mathsf{real}}_{conf,\mathcal{A}}(r)$ when r is chosen uniformly at random.

Ideal-world model: The ideal-world model that corresponds to a configuration $conf = (G(V,E), V^*, \mathcal{L}, \mathcal{C}_{node}, \mathcal{C}_{link})$ and adversary $\mathcal{A}$ is denoted by $sys^{\mathsf{ideal}}_{conf,\mathcal{A}}$, and it is illustrated on the right hand side of Figure 1. One can see that the ideal-world model is similar to the real-world model; the main difference is that machines M_i $(1 \leq i \leq n)$ are replaced with a new machine called T. The operation of the ideal-world model is very similar to the real-world model, therefore, we do not detail it here. We focus only on the operation of the new machine T.

In effect, machine T emulates the behavior of the machines M_i $(1 \leq i \leq n)$, with the difference that T is initialized with $conf$, and hence, it can detect when the system gets into an incorrect state. When this happens, T records that the system has been in an incorrect state, but the computation continues as if nothing wrong had happened.

Similar to the real-world model, the computation ends, when H reaches one if its terminal states, which happens when H receives a response to each of the requests that it placed on the tapes req_i $(1 \leq i \leq n)$, where a response can also be a time-out. The output of the ideal-world model is either the ensemble of the routing tables if T has not recorded an incorrect state during the computation, or a special symbol that indicates that an incorrect state has been encountered. The output is denoted by $Out^{\mathsf{ideal}}_{conf,\mathcal{A}}(r)$. Moreover, $Out^{\mathsf{real}}_{conf,\mathcal{A}}$ denotes the random variable describing $Out^{\mathsf{real}}_{conf,\mathcal{A}}(r)$ when r is chosen uniformly at random.

2.4 Definition of Security

Based on the model introduced in the previous subsections, we define routing security formally as follows:

Definition 2. *(Statistical security) A routing protocol is said to be statistically secure if, for any configuration conf and any real-world adversary $\mathcal{A}$, there exists an ideal-world adversary $\mathcal{A}'$, such that $Out^{\mathsf{real}}_{conf,\mathcal{A}}$ is statistically indistinguishable*[1] *from $Out^{\mathsf{ideal}}_{conf,\mathcal{A}'}$.*

The intuitive meaning of the definition above is that if a routing protocol is statistically secure, then any system using this routing protocol gets into an incorrect

[1] Two random variables are statistically indistinguishable if the L_1 distance of their distributions is negligibly small.

state only with negligible probability. This negligible probability is related to the fact that the adversary can always forge the cryptographic primitives (e.g., generate a valid digital signature) with a very small probability.

3 Insecurity of SAODV

SAODV [14] is a “secure” variant of the Ad hoc On-demand Distance Vector (AODV) [12] routing protocol. In the following, we briefly overview the operation of SAODV, and we show that, in fact, it is not secure in our model.

3.1 Operation of SAODV

The operation of SAODV is similar to that of AODV, but it uses cryptographic extensions to provide integrity of routing messages and to prevent the manipulation of the hop count information. Conceptually, SAODV routing messages (i.e., route requests and route replies) have a non-mutable and a mutable part. The non-mutable part includes, among other fields, the node sequence numbers, the addresses of the source and the destination, and a request identifier, while the mutable part contains the hop count information. Different mechanisms are used to protect the different parts.

The non-mutable part is protected by the digital signature of the originator of the message (i.e., the source or the destination of the route discovery). This ensures that the non-mutable fields cannot be changed by an adversary without the change being detected by the non-corrupted nodes.

In order to prevent the manipulation of the hop count information, the authors propose to use hash chains. When a node originates a routing message (i.e., a route reply or a route request), it first sets the HopCount field to 0, and the MaxHopCount field to the TimeToLive value. Then, it generates a random number *seed*, and puts it in the Hash field of the routing message. After that, it calculates the TopHash field by hashing *seed* iteratively MaxHopCount times. The MaxHopCount and the TopHash fields belong to the non-mutable part of the message, while the HopCount and the Hash fields are mutable. Every node receiving a routing message hashes the value of the Hash field (MaxHopCount − HopCount) times, and verifies whether the result matches the value of the TopHash field. Then, before rebroadcasting a route reply or forwarding a route request, the node increases the value of the HopCount field by one, and updates the Hash field by hashing its value once.

The rationale behind using the above hash chaining mechanism is that given the values of the Hash, the TopHash, and the MaxHopCount fields, anyone can verify the value of the HopCount field. On the other hand, preceding hash values cannot be computed starting from the value in the Hash field due to the one-way property of the hash function. This ensures that an adversary cannot decrease the hop count, and thus, cannot make a route appearing shorter than it really is. However, as we will see later (and as pointed out by the authors of SAODV themselves), this latter statement does not hold in general, because a corrupted node that happens to be on a route between the source and the destination may pass on the routing message without increasing the value of the HopCount field and without updating the value of the Hash field.

3.2 Simple Attacks Against SAODV

According to our definition of security, a routing protocol is secure if it ensures that incorrect entries in the routing tables of the non-corrupted nodes can be generated only with negligible probability. In case of SAODV, a node v creates an entry in its routing table for a target ℓ_{tar} only if it receives a fresh enough routing message that carries a valid digital signature of ℓ_{tar}. The fact that this routing message arrived to v means that there must be a route between v and a node that uses the identifier ℓ_{tar}, since otherwise, the message cannot reach v. However, SAODV cannot guarantee that the next hop and the hop count information in the newly created routing table entry is correct. This is illustrated by the following two examples.

Attack 1: Let us consider the configuration illustrated in Figure 2. Since SAODV uses the hop count as the routing metric, we set the node cost to 1 for every node and the link cost to 0 for every link. Let us assume that the node labelled by S starts a route discovery towards the node labelled by T. When the route request message reaches the corrupted node labelled by Z, it does not increase the hop count and does not update the hash value in the message. Therefore, when this route request is eventually received by the node labelled by T, it will create an entry $(S, B, 1)$ in its routing table. In addition, this entry will not be overwritten when the other route request message arrives through the node labelled by C, since that request will have a hop count of 2. This means that the system ends up in an incorrect state, because there is not any route in this network that starts at the node labelled by T, passes through the node labelled by B, ends at the node labelled by S, and has a cost less than or equal to 1.

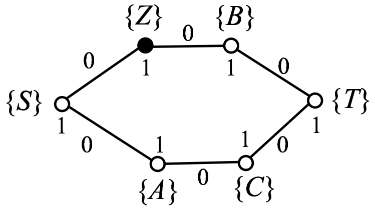

Fig. 2. A configuration where the adversary can achieve that the node labelled by T creates in its routing table an entry with an incorrect cost value when SAODV is used

We note that this weakness of SAODV has already been known by its authors (see Subsection 5.3.5 of [14]). Our purpose with this example is simply to illustrate how an attack that exploits the weakness can be represented within our framework.

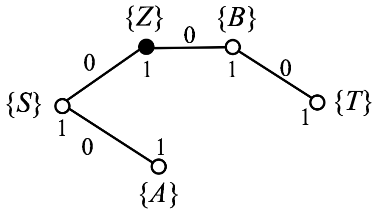

Fig. 3. A configuration where the adversary can achieve that the node labelled by S creates in its routing table an entry for target T with an incorrect next hop A when SAODV is used

Attack 2: Let us now consider the configuration illustrated in Figure 3. Let us assume again that the source is the node labelled by S and the destination is the node labelled by T. Furthermore, let us assume that a route request message reached the destination, and it returned an appropriate route reply. When this reply reaches the corrupted node labelled by Z, it forwards it to the node labelled by S in the name of A. Therefore, the node labelled by S will create a routing table entry $(T, A, 2)$. Note, however, that there is no route at all from the node labelled by S to the node labelled by T that passes through the node labelled by A. In other words, the system ends up in an incorrect state again. To the best of our knowledge, this weakness of SAODV has not been published yet.

4 Security of ARAN

ARAN (Authenticated Routing for Ad hoc Networks) is another secure, distance vector routing protocol for ad hoc networks proposed in [13]. In this section, we briefly overview its operation, and we prove that it is secure in our model.

4.1 Operation of ARAN

Just like SAODV, ARAN as well uses public key cryptography to ensure the integrity of routing messages. Initially, a source node S begins a route discovery process by broadcasting a route request message:

$$(\mathsf{RREQ}, T, cert_S, N_S, t, Sig_S)$$

where RREQ means that this is a route request, S and T are the identifiers of the source and the destination, respectively, N_S is a nonce generated by S, t is the current time-stamp, $cert_S$ is the public-key certificate of the source, and Sig_S is the signature of the source on all of these elements. N_S is a monotonically increasing value that, together with t and S, uniquely identifies the message, and it is used to detect and discard duplicates of the same request (and reply).

Later, as the request is propagated in the network, intermediate nodes also sign it. Hence, the request has the following form in general:

$$(\mathsf{RREQ}, T, cert_S, N_S, t, Sig_S, Sig_A, cert_A)$$

where A is the identifier of the intermediate node that has just re-broadcast the request. When a neighbor of A, say B, receives this route request, then it verifies both signatures, and the freshness of the nonce. If the verification is successful, then B sets an entry in its routing table with S as target, and A as next hop. Then, B removes the certificate and the signature of A, signs the request, appends its own certificate to it, and rebroadcasts the following message:

$$(\mathsf{RREQ}, T, cert_S, N_S, t, Sig_S, Sig_B, cert_B)$$

When destination T receives the first route request that belongs to this route discovery, it performs verifications and updates it routing table in a similar manner as it is done by the intermediate nodes. Then, it sends a route reply message to S. The route reply is propagated back on the reverse of the discovered route as a unicast message. The route reply sent by T has the following form:

$$(\mathsf{RREP}, S, cert_T, N_S, t, Sig_T)$$

where RREP means that this is a route reply, N_S and t are the nonce and the time-stamp obtained from the request, S is the identifier of the source, $cert_T$ is the public-key certificate of T, and Sig_T is the signature of T on all of these elements.

Similar to the route request, the route reply is signed by intermediate nodes too. Hence, the general form of the route reply is the following:

$$(\mathsf{RREP}, S, cert_T, N_S, t, Sig_T, Sig_B, cert_B)$$

where B is the identifier of the node that has just passed the reply on.

A node A that receives the route reply verifies both signatures in it, and if they are valid, then it forwards the reply to the neighbor node from which it has received the corresponding route request previously. However, before doing that, A will remove the certificate and the signature of B, and put its own certificate and signature in the message:

$$(\mathsf{RREP}, S, cert_T, N_S, t, Sig_T, Sig_A, cert_A)$$

In addition, A also sets an entry in its routing table for target T with B as the next hop.

As it can be seen from the description, ARAN does not use hop counts as a routing metric. Instead, the nodes update their routing tables using the information obtained from the routing messages that arrive first; any later message that belongs to the same route discovery is discarded. This means that ARAN may not necessarily discover the shortest paths in the network, but rather, it discovers the quickest ones. In effect, ARAN uses the message propagation delay (i.e., physical time) as a path length metric.

4.2 Security Proof

Theorem 1. *ARAN is a secure ad hoc routing protocol in our model, if the signature scheme is secure against chosen message attacks.*

Proof. Since ARAN uses the message propagation delay as the routing metric, we will assume that the node cost values in our model represent minimum message processing delays (at the nodes), and the link cost values represent minimum message transmission delays (on the links). In addition, we make the pessimistic assumption that the adversary's message processing delay is 0, which means that $\mathcal{C}_{node}(v) = 0$ for all $v \in V^*$.

In order to be compliant with our framework, we also assume that each routing table entry explicitly contains a routing metric value too. In our case, this metric value is the time that was needed for the routing message that triggered the creation of this entry to get from the originator of the message to the node that created this entry. Although these times are not represented explicitly in ARAN routing table entries, representing them in the model does not weaken our results in any way. In particular, exactly the same routing table entries are created in our model as in ARAN with respect to the target and the next hop identifiers.

In order to prove that ARAN is secure, one has to find the appropriate ideal-world adversary $\mathcal{A}'$ for any real-world adversary $\mathcal{A}$ such that Definition 2 is satisfied. Due to the constructions of our models, a natural candidate is $\mathcal{A}' = \mathcal{A}$, since in that case, the steps of the real-world and the ideal-world models are exactly the same (for the same random input, of course). If no incorrect state is encountered during the computation in the ideal-world model, then not only the steps, but the outputs of the two models will be the same too. On the other hand, if an incorrect state occurs in the ideal-world model, then the outputs of the models will be different, since the ideal-world model will output a special symbol. Hence, Definition 2 is satisfied, if an incorrect state can only be encountered with negligible probability. We will show that indeed this is the case for ARAN.

Getting into an incorrect state means that one of the non-corrupted nodes, say v, sets an incorrect entry in its routing table. Let this incorrect entry be $(\ell_{tar}, \ell_{nxt}, c)$. Since v is non-corrupted, it sets this entry only if it received a routing message that has been signed by ℓ_{tar} as originator and ℓ_{nxt} as previous hop, v has a neighbor that uses identifier ℓ_{nxt}, and it took time c for the message to get from its originator to v. Now, $(\ell_{tar}, \ell_{nxt}, c)$ can be incorrect for one of the following three reasons:

1. There is no route from v to a node that uses ℓ_{tar}.
2. There are routes from v to a node that uses ℓ_{tar}, but none of them go through any neighbor of v that uses ℓ_{nxt}.
3. There are routes from v to a node that uses ℓ_{tar} going through a neighbor of v using ℓ_{nxt}, but each of them has a cost higher than c.

In case 1, if the signature of ℓ_{tar} in the routing message is not forged, then the very fact that v received the message proves that there is a route between v and a node that uses ℓ_{tar} (since otherwise the message could not reach v). Hence, case 1 is possible only if the signature of ℓ_{tar} is forged, and this has negligible probability if the signature scheme is secure.

In case 2, if the signature of ℓ_{nxt} in the routing message is not forged, then a neighbor of v, say v', that uses ℓ_{nxt} has indeed seen and signed the message. Now, the same reasoning can be used for v' as in case 1 for v: if the signature of ℓ_{tar} in the routing message is not forged, then the fact that v' received the message proves that there is a route between v' and a node that uses ℓ_{tar}, and hence, there is a route between v and a node that uses ℓ_{tar} that goes through v' (since v' is a neighbor of v). This means that case 2 is possible only if the signature of ℓ_{tar} or ℓ_{nxt}, or both are forged, and this has negligible probability.

Finally, in case 3, let R be the set of existing routes that start at v, end at a node that uses ℓ_{tar}, and go through a neighbor of v using ℓ_{nxt}. Moreover, let c' be the minimum of the costs of the routes in R. By assumption, $c' > c$. If the signatures of ℓ_{tar} and ℓ_{nxt} in the routing message received by v are not forged, then the message must have taken one of the routes in R. However, it could not reach v in time $c < c'$, since the node and link costs represent the minimum message processing and transmission delays at the nodes and on the links. In other words, the adversary cannot speed up the transmissions on the links and the processing at the non-corrupted nodes. Hence, case 3 is possible only if either ℓ_{tar} or ℓ_{nxt}, or both are forged, which can happen only with negligible probability. ■

5 Related Work

There are several proposals for secure ad hoc routing protocols (see [7] for a recent overview). However, most of these proposals come with an informal security analysis with all the pitfalls of informal security arguments. Another set of related papers deal with provable security for cryptographic algorithms and protocols (see Parts V and VI of [9] for a survey of the field) and with the application of formal methods for the security analysis of cryptographic protocols (see [4] for an overview of the main approaches). However, these papers are not concerned with ad hoc routing protocols. There exist only a few papers where formal techniques are proposed for the verification of the security of ad hoc routing protocols; we briefly overview them here.

In [15], the authors propose a formal model for ad hoc routing protocols that is similar to the strand spaces model [5], which has been developed for the formal verification of key exchange protocols. Routing security is defined in terms of a safety and a liveness property. The liveness property requires that it is possible to discover routes, while the safety property requires that discovered routes do not contain corrupted nodes. In contrast to this, our definition of security admits routes that pass through corrupted nodes, because it seems to be impossible to guarantee that discovered routes do not contain any corrupted node, given that corrupted nodes can behave correctly and follow the routing protocol faithfully.

Another approach, presented in [10], is based on a formal method, called CPAL-ES, which uses a weakest precondition logic to reason about security protocols. Unfortunately, the work presented in [10] is very much centered around the analysis of SRP [11], and it is not general enough. We must also mention that in [11], SRP has been analyzed by its authors using BAN logic [2]. However, BAN logic has never been intended for the analysis of routing protocols, and there is no easy way to represent the requirements of routing security in it. In addition, a basic assumption of BAN logic is that the protocol participants are trustworthy, which does not hold in the typical case that we are interested in, namely, when there are corrupted nodes in the network controlled by the adversary that may not follow the routing protocol faithfully.

Finally, in [3] and [1], we have developed and applied an approach based on the simulation paradigm for on-demand source routing protocols for ad hoc networks. The framework proposed in this paper is essentially the adaptation of that approach to on-demand, distance vector routing protocols.

6 Conclusion

In this paper, we proposed an approach for the security analysis of on-demand, distance vector routing protocols for ad hoc networks, such as AODV, SAODV, and ARAN. The proposed approach is based on the simulation paradigm that is used extensively for the analysis of cryptographic algorithms and protocols, and it provides a rigorous method for proving that a given routing protocol is secure. We demonstrated the approach by representing two attacks on SAODV in our framework, and by proving that ARAN is secure in our model.

An important message of this paper is that flaws (leading to attacks) in ad hoc routing protocols can be very subtle, and hard to discover by informal reasoning. Another important message is that it is possible to adopt sound analysis techniques known from the cryptographic literature, and to use them in the context of ad hoc routing protocols.

In our future work, we intend to automate (at least partially) the process of the security analysis of ad hoc routing protocols. For this purpose, we will identify an appropriate formal framework, e.g., one based on model checking. Furthermore, our current definition of a correct state is not strict enough, because it does not consider that an adversary might have an interest in increasing the cost of a route passing through it (perhaps, to get rid of the traffic). Thus, we intend to extend the definition of a correct routing table entry by requiring an appropriate upper bound on the believed cost of the route.

Acknowledgements

The work presented in this paper has partially been supported by the Hungarian Scientific Research Fund (T046664). The second author has been further supported by IKMA and by the Hungarian Ministry of Education (BÖ2003/70).

References

[1] G. Ács, L. Buttyán, and I. Vajda. Provably Secure On-demand Source Routing in Mobile Ad Hoc Networks. Technical Report, Budapest University of Technology and Economics, March 2005. Available on-line at: http://www.hit.bme.hu/˜buttyan/publications.html

[2] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, February 1990.

[3] L. Buttyán and I. Vajda. Towards provable security for ad hoc routing protocols. In *Proceedings of the ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN)*, October 2004.

[4] R. Focardi and R. Gorrieri (eds). *Foundations of Security Analysis and Design.* LNCS 2171, Springer-Verlag, 2000.

[5] J. Guttman. Security goals: packet trajectories and strand spaces. In *Foundations of Security Analysis and Design*, edited by R. Focardi and R. Gorrieri, Springer LNCS 2171, 2000.

[6] Y.-C. Hu, A. Perrig, and D. Johnson. Ariadne: A secure on-demonad routing protocol for ad hoc networks. In *Proceedings of the ACM Conference on Mobile Computing and Networking (Mobicom)*, 2002.

[7] Y.-C. Hu and A. Perrig. A survey of secure wireless ad hoc routing. *IEEE Security and Privacy Magazine*, 2(3):28–39, May/June 2004.

[8] D. Johnson and D. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, edited by Tomasz Imielinski and Hank Korth, Chapter 5, pages 153–181. Kluwer Academic Publisher, 1996.

[9] W. Mao. *Modern Cryptography: Theory and Practice.* Prentice Hall PTR, 2004.

[10] J. Marshall. An Analysis of the Secure Routing Protocol for mobile ad hoc network route discovery: using intuitive reasoning and formal verification to identify flaws. MSc thesis, Department of Computer Science, Florida State University, April 2003.

[11] P. Papadimitratos and Z. Haas. Secure routing for mobile ad hoc networks. In *Proceedings of SCS Communication Networks and Distributed Systems Modelling Simulation Conference (CNDS)*, 2002.

[12] C. Perkins and E. Royer. Ad hoc on-demand distance vector routing. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, pp. 90-100, February 1999.

[13] K. Sanzgiri, B. Dahill, B. Levine, C. Shields, and E. Belding-Royer. A secure routing protocol for ad hoc networks. In *Proceedings of the International Conference on Network Protocols (ICNP)*, 2002.

[14] M. G. Zapata and N. Asokan. Securing ad hoc routing protocols. *Proceedings of the ACM Workshop on Wireless Security (WiSe)*, 2002.

[15] S. Yang and J. Baras. Modeling vulnerabilities of ad hoc routing protocols. In *Proceedings of the ACM Workshop on Security of Ad Hoc and Sensor Networks*, October 2003.