Analysis of SuperSEAD: A secure Ad Hoc Routing Protocol.

Team: Aaron Staple, Mukund Sundararajan.

Abstract: This write up contains the details of the analysis of a secure routing protocol, using Murϕ, a state machine checker. The analysis strategy uses an equivalence notion of security. It considers the comparison of the secure protocol in the presence of attackers with its un-secured counterpart, in the absence of attackers.

There are a few novel ways that Murϕ has been used in this analysis, including modifying the Murϕ source to have it output all the deadlocked states.

We conclude by presenting our views on the protocol that we analyzed and a few suggestions features that a tool to analyze such protocols must contain.

Introduction:

SuperSEAD is a secure routing protocol, intended for use in mobile ad-hoc networks. It uses hash chains and hash tree chains in various ways to achieve correct routing state in the presence of multiple uncoordinated attackers. The version that we analyzed is described in [1].

We picked this protocol to analyze, as it seemed to be sufficiently different from the kinds of protocols described in class, to offer us some fundamental challenges.

We picked Murϕ to analyze the protocol as we were studying the effect of attackers advertising wrong routing state, and the Murϕ notion of states, transitions and invariants sufficed for out analysis.

To see what we did, see the section on Analysis Process.

Ad Hoc Network Characteristics:

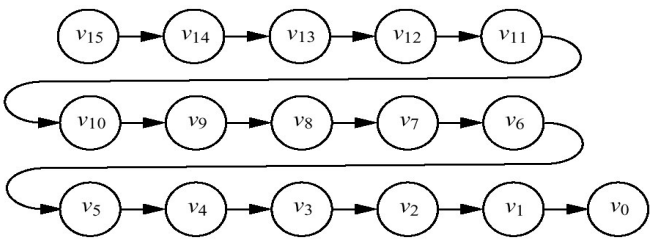
These networks are characterized by need for low power consumption and low levels of physical security and broadcast physical medium.

Asymmetric techniques like RSA are not to be used as are inefficient and consume too much power.

Theory:

The SuperSEAD protocol consists of a few, repeating security patterns, this section contains a brief description of these security patterns.

Hash chains:



Hash Chains are used to provide authentication in situations that demand non-resource intensive alternatives to the heavy, asymmetric or symmetric crypto based methods.

An arbitrary starting value is picked and each element in the chain is a hash of the previous element. In the diagram above, v_{15} is the arbitrarily picked initial value. The remaining values are generated using a 'well known' hash function.

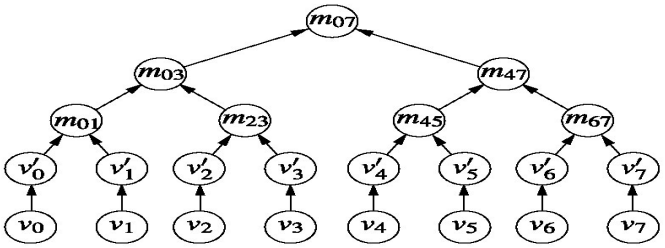
The anchor of the hash chain, v_0 is distributed to the entities that intend to authenticate the sender.

The sender sends out hash values starting v_1 onwards in successive packets that it sends out. Receivers can authenticate these packets by hashing the received value an appropriate number of times and checking whether the result is the same as a previously authenticated value (or the anchor).

The basis for security relies on the fact that an attacker cannot forge an element in that is earlier in the generation order based on a known value in the hah chain. This property is realized by the collision free nature of hash functions.

Hash Trees:

These are generalizations of hash chains. Hash trees allow the anchor(the root of the tree) to authenticate the separately, any leaf of the tree.



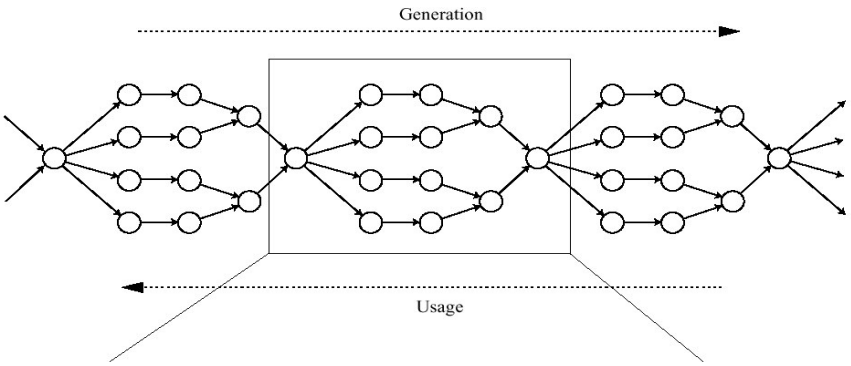
Generation starts from the leaf nodes, which are hashed once to blind the values.
Each pair of sibling nodes are concatenated and hashed to form a parent. This process is continued upwards to the root.
The anchor (the root) is distributed initially to all receivers that intend to authenticate the packets. A sender sends a specific leaf node along with a path up the tree that allows the receiver to authenticate the sender. For example a sender could send v_4, v_5', m_{67}, m_{03} and the receiver could the repeat the generation procedure and check it against the anchor.

Hash Tree Chains:

These are compositions of hash chains and hash trees.

Here each node in the hash tree chain is a hash tree. Successive hash trees are generated by concatenating the root with 1,2,…k and hashing each concatenation. Where k is the number of leaf nodes in the tree.

The generation and authentication procedures are similar to hash chains.



Packet Leashes: These are used to avoid replay attacks. The leashes come in two flavors, geographical leashes and temporal leashes.
Leashes disallow attacks that make the receiver believe that a sender is within one hop when it is not.

Temporal Leashes work as follows: The sender sends a timestamp signifying the time of dispatch of the packet. An authentication scheme is used to allow the receiver to verify the authenticity of the timestamp. The receiver can check that the packet was generated not too far back in the past, and thus verify that there was no replay attack.
Temporal Leashes require that the clocks of the sender and the receiver are tightly synchronized.

Geographical Leashes: Geographical Leashes are similar to temporal leashes with the additions that the sender also publishes his location information.
The Receiver can verify the fact that the sender is within range, by comparing the senders location with its own location, to ensure that the transmission was made within the locality of the receiver.
Geographical Leashes require that each node knows it own location.

Tesla: This is a broadcast authentication protocol that is based on hash chains and time synchronization.
The Basic Idea is as follows: All the nodes have loosely synchronized clocks. Each node generates a hash chain for itself and distributes the anchor to all the nodes that intend to authenticate its broadcasts.
Time is divided into intervals, and hash chain nodes are associated with each interval in reverse generation order.

The Sender then includes the corresponding hash chain value with all broadcasts made during that interval, and then reveals these values according to a previously established schedule. Receivers Buffer packets till they receive the hash chain values that allow them to authenticate the packets.

Security arises from the fact that the hash value that is used to authenticate all broadcasts during the interval is not published till a later interval.

This results in an authentication scheme that relies on the relatively resource non-intensive hash chains.

Assumptions that SuperSEAD makes:

1. Attacks that rely on hiding certain routes are not guarded against, as this situation is equivalent to not forwarding data packets.
2. Nodes cannot selectively jam packets as these attacks originate at the physical layer and require mechanisms in the physical layer to avoid them.
3. Attacker nodes cannot collude to attack the network.
4. Nodes can replay, modify routing information
5. When a node is compromised all the key material that is at the node is available to the attacker.

Protocol Description:

SEAD is based on a periodic distance vector routing protocol DSDV, with some modifications.

The routing protocol is as follows:
Each node sends out a periodic update about itself with an even sequence number. It includes in each update, information about every other node: distance, next hop, and the sequence number of the last update that it received about the node.

A node that receives a routing update from a neighbor updates an entry in the routing table about another node if the update has a higher sequence number or the same sequence number and a smaller metric value.

When a the neighbor of a node moves, the node detects this (through physical layer mechanisms) and marks the corresponding routing table entry with the next odd sequence number. It thus avoids the count to infinity problem by ignoring updates, till the node which move publishes its next even sequence numbered update.

"SEAD is robust against multiple uncoordinated attackers creating incorrect routing state in any other node, even in spite of any active attackers or compromised nodes in the network."

It achieves this security using the following ideas:

1. **Hash chains and Hash tree chains** to authenticate higher sequence numbers and a lower bound on the metric value.
This is implemented as follows: Each node generates a chain of length m*k, where m is the number of nodes in the network (and a bound on the maximum distance between any two nodes on the network.), and k is the maximum sequence number.

The anchor of this chain is distributed to all the nodes that intend to authenticate this node.

A node would include a fresh sequence number and a zero metric value about itself in the update that it sends to its neighbors.

Each good node, would check the update that it receives and see if it has a higher sequence number or a similar sequence number and a lower metric. In which case it would use the advertising node as the next hop and make necessary modifications to the routing table entry.

A good node advertises to its neighbors a distance of one hop more than what it has in its routing table to any destination.

A tuple (dist,seqno) for a particular destination is represented in the hash chain by the element corresponding to (k-seqno)*m +dist in the generation order of the hash chain.

Incrementing the metric is reflected in the hash chain by performing a single hash.
Note that a bad node could still advertise the same metric that they receive.

Verification of the authenticity of an update is ensured by hashing the received hash value an appropriate number of times and checking it against a previously authenticated hash value (or the anchor).

2.Packet leashes along with Hash tree chains to avoid the same distance attack.

Hash tree chains tie the authenticator with the sending node id, and along with packet leashes ensure that bad nodes cannot advertise the same metric that they receive.
Each node corresponds to a different leaf node in the hash-tree. The packet leashes ensure that a node can detect that the node that sent it the packet is really its neighbor. Since each node has to specify a different path up the hash- tree, it cannot advertise the same metric and needs to follow the hash tree structure to the next level.

3.Neighbour Authentication to ensure that the advertiser of the information is authentic.
This is because SEAD could potentially select the next-hop to be the advertiser.
This authentication is achieved via TESLA or by having a key per pair of nodes.
This authentication is redundant if we have packet leashes.

Analysis Process:

An equivalence based approach:

The security criterion that we are trying to check is that SEAD maintains correct routing state at the good nodes in the network in the presence of multiple un-coordinated attackers.

A natural formulation of the checking is to compare the secure version of the Routing protocol in the presence of attackers against the un-secured version in the absence of attackers. What this does is that it allows checking precisely what the secure version attempts to achieve which is equivalence to the non-secure version without any attackers.

We think that the equivalence-based method allows easy formulation of the security criteria.

We chose not to compare the two protocols at all steps, but only in the steady states of the network.

These steady states are defined by all the states where the good nodes cannot make any moves.

We printed out all the steady states that the protocol reached by hacking Murϕ to output all the states in which the good nodes had no further moves.

A typical scenario was to simulate topology changes (as described in the next section) and check whether the final routing state achieved was the same as the case in the absence of attackers.

We used the all-pairs shortest path algorithm to find the routing state achieved in the un-secured scenario.

Mixture of simulation and verification:

Ideally we would have liked to run verification for a large number of nodes with a fraction of the nodes being attacker nodes. We would also have liked to enumerate all possible connectivities.

The state space however was very large. So the idea that we adopted was to simulate sequence of random topologies, and verify the protocol against all possible attacker moves.

Simulating changing topology:

We use a C program to generate a sequence of topologies ,where each successive topology has exactly one node moved to a new place. That is it differs from the previous topology in one row and one column.

Simulating the Hash chains:

We assumed a Dolev-Yao model, i.e. the hashing is hard to invert. Thus the moves that we simulated for the attacker always caused it to advertise higher hash chain(in the gene members than it received. To be precise, lower sequence numbers or equal sequence numbers and higher metrics.

The difference between SEAD and SuperSEAD is the fact that in SEAD a bad node can advertise the same metric for a particular destination that it has received about that destination. SuperSEAD uses hash tree chains to guard against this situation

We assumed that the neighbor authentication and packet leashes are present.

Attack Scenarios that we inspected:

1.We ran Murϕ for the case with the packet leashes, hash chains present and found no attack.

2.We ran the protocol without the hash chain present and found the obvious attack.

That is the bad nodes cause incorrect routing state by advertising arbitrary distance values.

3.We also simulated a tunnel scenario, where two bad nodes could share the routing information, essential simulating routing information being tunneled as data.
We found a wormhole attack in this case.

4.We also noted if the protocol used geographical packet leashes, in allows the attackers to use this information to place themselves at specific spots in the network to form a cut, of the network.

5.We simulated a situation where even in the presence of packet leashes and hash chain trees, if there are k(>1) bad nodes on path then they can reduce the length of the path by k-1.

Challenges Faced:

1.We found it hard to specify what the security invariant should be, and at what states we should check it.
It was interesting to note that the concept of a deadlock actually came in handy during the analysis of a security protocol.
2. We ran into trouble with an inconsistent specification of the Murϕ grammar.
The non-terminal 'quantifier' is specified as follows:
<quantifier> ::= <ID> : <typeExpr>
| <ID> := <expr> to <expr> [by <expr>]

The second production does not function within a rule-set. A work around for this to use the following production instead:
<quantifier> ::=
| <ID> : expr to expr

3.It would be nice to have a formal protocol description language that is universally accepted, requiring that all new protocols be specified in this language. It would simplify the communication of protocols.

Conclusions regarding SuperSEAD:

1.SEAD makes the does not allow for new nodes to join the network.
In MANET this seems to be an unrealistic requirement.

2.It is unable to defend against a simple collusion attack, involving two attacker nodes sharing information via a tunnel.(Sending the routing information as data).
This attack can be used to achieve a wormhole between the two bad nodes. This causes a lot of communication to pass through them. This situation can then be exploited further to achieve further attacks.

3.The protocol description for SEAD was in prose, and we took a number of decisions based on our understanding of what the authors meant.

4.The protocol uses hash chains and related constructs very effectively for authentication. These constructs would also satisfy the low power requirement.

5.The protocol does not guard against denial of service attacks. The authors justify this by saying that such attacks begin at the physical layer and should be dealt with there.
This justification is not completely reasonable as denial of service is possible at the network layer even without any physical jamming of signals.

Conclusions regarding Tools:

1.Murϕ should allow a mixture of simulation and verification methods. This allows the user can perform state space reduction in a natural manner by marking certain rules for simulation and others for verification.

We do not think that there would be any fundamental difficulty in trying to modify Murϕ in trying to accomplish this.

2.It would be nice to have a channel abstraction in Murϕ, with a specification of all the entities that could listen in on the channel, in terms of a connectivity matrix.
This might simplify the modeling of attacker knowledge, in the case of weaker attackers.

3. Also some of the trace sequences that we read were extremely long, and it might help introducing 'program slicing' ideas to printing traces.

This can be done as follows: A set of variables that represents a subset of the entire state space is specified and a the trace printed contains only states that differ in one of these variables from the previous state.

References:

1. SEAD: secure efficient distance vector routing for mobile wireless ad hoc networks.
Yih-Chun Hu,David B.Johnson,Adrian Perrig

2. Y.-C.Hu, A.Perrig, D.B.Johnson, Packet leashes: a defense against wormhole attacks in wireless ad hoc networks, in: Proceedings of IEEE Infocomm 2003,April 2003.

3. Efficient Security Mechanisms for Routing Protocols.
Yih-Chun Hu ,Adrian Perrig,David B. Johnson

4. Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures.
Chris Karlof David Wagner

5. The TESLA Broadcast Authentication Protocol .
Adrian Perrig Ran Canetti J. D. Tygar Dawn Song