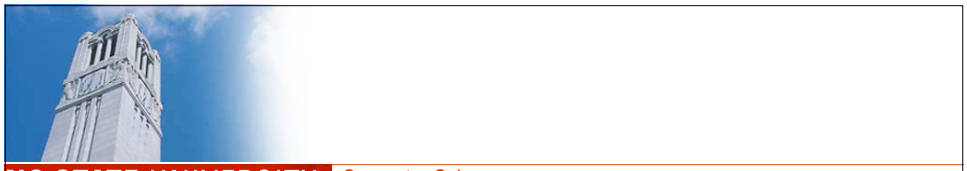


NC STATE UNIVERSITY Computer Science

Ariadne – A Secure On-Demand Routing Protocol for Ad-Hoc Networks

Authors: Yih-Chun Hu, Adrian Perrig, David B Johnson

Presenter: Sameer Korrapati

Date: 4/21/2003

Overview of presentation

- Introduction : Ad-hoc routing security issues & attacks, basic DSR mechanism
- Related work
- Contributions in this paper
- Ariadne details: Route-discovery, route-maintenance, node misbehavior, malicious floods
- An optimization for Ariadne
- Ariadne Evaluation & Security Analysis (if time permits)
- Conclusions & future work
- *Knowledge of TESLA assumed, not explained

Introduction

- Security in routing (not just data traffic) is a major concern as it can potentially bring down the whole network (attacks later)
- On demand routing is more suitable for ad-hoc networks as against periodic-updates based routing protocols (being *reactive* as opposed to *proactive*)
- Ariadne proposes a new *on-demand, secure* routing protocol for ad-hoc networks

Introduction-contd.

- Based on Dynamic Source Routing protocol (DSR)
- Authentication scheme has been explained with TESLA, and symmetric cryptography has been used in all messages

Attacks on routing in ad-hoc networks

- Active and Passive
- Selfishness and Maliciousness
- Some active attacks: *black hole, resource consumption, worm hole,* and many more.
- Attacker Model: Active-n-m ➔ n is the number of nodes attacker has compromised, m is the number of attacker's own nodes
- Ariadne provides resilience against Active-1-x and Active-y-x attacks

Overview of DSR

- Route Discovery Phase: *Initiator* broadcasts RREQ, if route not in cache, requesting for *target.* Each intermediate node appends its id and rebroadcasts. *target* sends RREP, including the list. – Source routing
- Route Maintenance Phase: Needed as routes can break over time. Node detecting this will send a ROUTE_ERROR to remove that route from cache.

Related Work

- Prevention based schemes:
  - SAODV (digital sign),
  - ARAN (certificate-based, with a TTP)
  - SEAD(one-way hash chains),
  - Packet Leashes (deals worm-hole attacks),
  - etc…
- Detection and reaction based:
  - Watchdog-and-Pathrater,
  - CONFIDANT (detects misbehaving nodes)
  - etc…

Contributions in this paper

- Proposed a complete secure routing protocol, based on on-demand philosophy.
- The security scheme made efficient using symmetric key MACs and hash chains
- Performance evaluation and security analysis presented.
- Explained some attacks on ad-hoc network routing.

Assumptions

- Key setup is crucial, whichever scheme is used, of the three. For TESLA, shared pair-wise secret keys between source and destination, (why?) and one public key needed (why?), apart from TESLA keys.
- Nodes strong enough to do needed computations and should have loose clock synchronization
- Each node can estimate end-to-end transmission time to all other nodes in the network (for TESLA)

Assumptions – contd.

- Only network layer security addressed. (Not MAC or physical layer, which are important in wireless ad hoc environments)
- Broadcast transmission medium is reliable

Ariadne Details

- Ariadne Route Discovery
- Ariadne Route Maintenance
- Thwarting routing misbehavior
- Thwarting malicious RREQ floods
- An optimization for Ariadne with TESLA
- Performance Evaluation
- Security Analysis

Notations

- Notations
  - A, B : principals
  - K_{AB}, K_{BA} : secret MAC keys between A and B
  - MAC (M) : MAC of message M using MAC key K_{AB}

Route Discovery

- 3 ways to authenticate: Digital signs, symmetric key MACs, TESLA.
- Use of TESLA has been explained, because of its efficiency.
- Goals:
  - ➢ Target can authenticate initiator
  - ➢ Initiator can authenticate each entry of the path
  - ➢ No intermediate node can remove a previous node's entry in the node list (per-hop hashing)

Route Discovery Authentication

- Authentication of RREQ by target: Initiator includes a MAC computed by shared-key K_{SD} with target
- Mechanisms for authenticating data in RREQs and RREPs: The appended node-lists have to be authenticated; Uses TESLA for efficiency. Intermediate nodes append their MACs computed with TESLA keys
- Per-hop hashing technique: To avoid a node from being removed from the node list in the RREQ message, a one-way hash function is used. The source initializes the hash chain to a MAC with a key shared between the source and target.

Route Discovery (contd.)

- Route Request
  - <RREQ, *initiator*, *target*, *id*, *time interval*, *hash chain*, *node_list*, *MAC_list*>
  - Initiator initializes *hash chain* to MAC_{KSD}(*initiator*, *target*, *id*, *time interval*)
  - Intermediate node *A* which receives the request checks <*initiator*, *id*> and checks *time interval*
    - *Time interval* : TESLA time interval at the pessimistic expected arrival time of the RREQ at target (say *T + 2d)*
    - If any condition fails, discard the request

Route Discovery (contd.)

- If all conditions hold, *A* appends its address to *node list*, replaces *hash chain* with *H*[*A*, *hash chain*], appends MAC of entire Request with TESLA key K_{A_i} to *MAC list*

  - Target checks validity of Request
    - By determining that the keys are not disclosed yet
    - that the *hash chain* is equal to $H[n_n, H[n_{n-1}, H[\ldots,H[n_1, MAC_{K_{SD}}(initiator, target, id, interval)]\ldots]$
    - If Request is valid, target returns a Route Reply

Route Discovery (contd.)

- Route Reply
  - <RREP, *target*, *initiator*, *time interval*, *node list*, *MAC list*, *target MAC*, *key list*>
  - Packet is sent to initiator along the route in *node list*
  - Forwarding node waits until it can disclose its key and then append its key
  - Initiator verifies that
    - Each key is valid (TESLA security condition)
    - *target MAC* is valid (based on K_{DC} shared with target)
    - Each MAC in *MAC list* is valid (based on TESLA keys)

Route Discovery example

$$
\begin{array}{ll}
S: & h_0 = \text{MAC}_{K_{SD}}(\text{REQUEST}, S, D, id, ti) \\
S \rightarrow *: & \langle \text{REQUEST}, S, D, id, ti, h_0, (), () \rangle \\
 & \\
A: & h_1 = H[A, h_0] \\
 & M_A = \text{MAC}_{K_{A_{ti}}}(\text{REQUEST}, S, D, id, ti, h_1, (A), ()) \\
A \rightarrow *: & \langle \text{REQUEST}, S, D, id, ti, \underline{\boldsymbol{h_1}}, (\underline{\boldsymbol{A}}), (\underline{\boldsymbol{M_A}}) \rangle \\
 & \\
B: & h_2 = H[B, h_1] \\
 & M_B = \text{MAC}_{K_{B_{ti}}}(\text{REQUEST}, S, D, id, ti, h_2, (A, B), (M_A)) \\
B \rightarrow *: & \langle \text{REQUEST}, S, D, id, ti, \underline{\boldsymbol{h_2}}, (A, \underline{\boldsymbol{B}}), (M_A, \underline{\boldsymbol{M_B}}) \rangle \\
 & \\
C: & h_3 = H[C, h_2] \\
 & M_C = \text{MAC}_{K_{C_{ti}}}(\text{REQUEST}, S, D, id, ti, h_3, (A, B, C), (M_A, M_B)) \\
C \rightarrow *: & \langle \text{REQUEST}, S, D, id, ti, \underline{\boldsymbol{h_3}}, (A, B, \underline{\boldsymbol{C}}), (M_A, M_B, \underline{\boldsymbol{M_C}}) \rangle \\
 & \\
D: & M_D = \text{MAC}_{K_{DS}}(\text{REPLY}, D, S, ti, (A, B, C), (M_A, M_B, M_C)) \\
D \rightarrow C: & \langle \text{REPLY}, D, S, ti, (A, B, C), (M_A, M_B, M_C), \underline{\boldsymbol{M_D}}, () \rangle \\
 & \\
C \rightarrow B: & \langle \text{REPLY}, D, S, ti, (A, B, C), (M_A, M_B, M_C), M_D, (\underline{\boldsymbol{K_{C_{ti}}}}) \rangle \\
 & \\
B \rightarrow A: & \langle \text{REPLY}, D, S, ti, (A, B, C), (M_A, M_B, M_C), M_D, (K_{C_{ti}}, \underline{\boldsymbol{K_{B_{ti}}}}) \rangle \\
 & \\
A \rightarrow S: & \langle \text{REPLY}, D, S, ti, (A, B, C), (M_A, M_B, M_C), M_D, \\
 & \quad (K_{C_{ti}}, K_{B_{ti}}, \underline{\boldsymbol{K_{A_{ti}}}}) \rangle
\end{array}
$$

Route Maintenance

- Need: In the source routes, an intermediate node might discover that next hop is not working. So it sends ROUTE_ERROR back to initiator
- Security issue: Prevent unauthorized nodes from sending (bogus) ROUTE_ERRORs
- The case of malicious nodes not sending genuine ROUTE_ERRORs not considered.

Route Maintenance (contd.)

- Route Error
  - <ROUTE_ERROR, *sending address*, *receiving address*, *time interval*, *error MAC*, *recent TESLA key*> source routed back to initiator
  - Intermediate node
    - Forwards the packet and searches its route cache for all routes that use <*sending address*, *receiving address*>
    - If exists, checks validity of *time interval*
    - If valid, checks authentication of the Error
    - Until authentication, saves Error info in memory until a key is disclosed and uses routes in route cache
    - If authenticated, removes all such routes

Thwarting routing misbehaviors

- Issue: What if intermediate nodes in the source route misbehave and don't forward packets?
  - A feedback based reputation scheme to detect misbehaving nodes - relies on feedback about which packets were successfully delivered
  - A node with multiple routes sends a fraction along each route and sends packets along the successful route
  - Malicious node avoidance in Route Discovery
    - Route Request includes a list of malicious nodes to avoid and the MAC h_0 computed over that list – no details
    - Malicious nodes can be detected by target

Thwarting malicious RREQ floods

- As flooded RREQs are authenticated at the target, and not at every hop, attacker can flood malicious RREQs.
- *Route Discovery chains*
  - To authenticate RREQs instantly
  - One-way chains generated as $K_i = H^{N-i}[K_N]$
  1. Release one key for each Discovery (rate-limiting requests)
     - A node not partitioned from the network can prevent an attacker from reusing keys
  2. Dictate schedule for disclosure of key + loose clock synchronization (time-synchronizing chain elements)
     - Any node can prevent an attacker from reusing keys
     - Computationally slightly more expensive

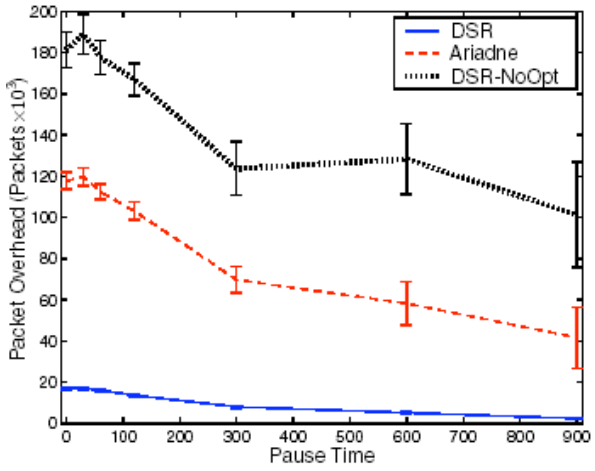
An optimization

- So far, only initiator can authenticate the target and its RREP, using the MAC computed with K_{DS}
- But this RREP passes all intermediate nodes, which can also use it, *if authenticated.*
- Solution: Target uses TESLA key that can be used by these intermediate nodes to authenticate it.
- The key is released by target, to initiator, after appropriate delay, which the intermediate nodes also use.

Performance Evaluation

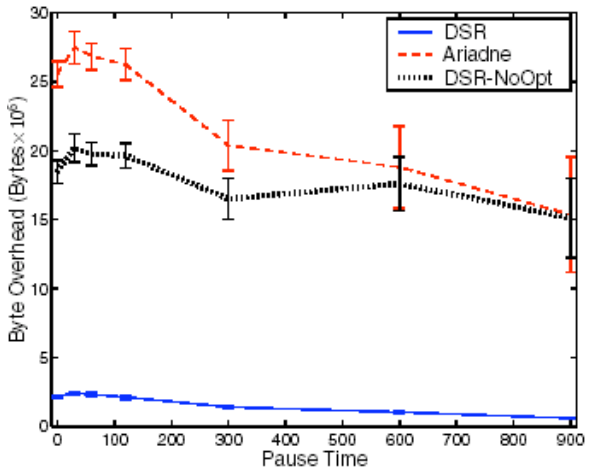
- ns-2 simulator has been used and following studied:
  - Packet Delivery Ratio
  - Packet overhead
  - Byte overhead
  - Mean Latency
  - 99.99^{th} percentile latency
  - Path optimality

Performance Evaluation – packet overhead



(b) Packet Overhead

Performance Evaluation – byte overhead



(c) Byte Overhead

» 26.19% higher byte overhead for Ariadne compared to un-optimized DSR because of authentication overhead

Security Analysis

- Argue that if an uncompromised shortest path exists, then Ariadne will find it (correctness)
- In case of Active-1-1 attacker, black holes and gray holes could be thwarted. (per-hop hashing), flooding of RREQs could be thwarted (route discovery chains)
- Passive attacks (eavesdropping) cannot be dealt with

Conclusions & future work

- Ariadne adds security extensions to on-demand routing (DSR), which suits ad-hoc networks well.
- Shown to be better than un-optimized DSR
- Ariadne prevents a wide range of attacks to secure routing
- It is efficient in terms of communication and computational overhead.
  - ➢Uses MACs and TESLA