

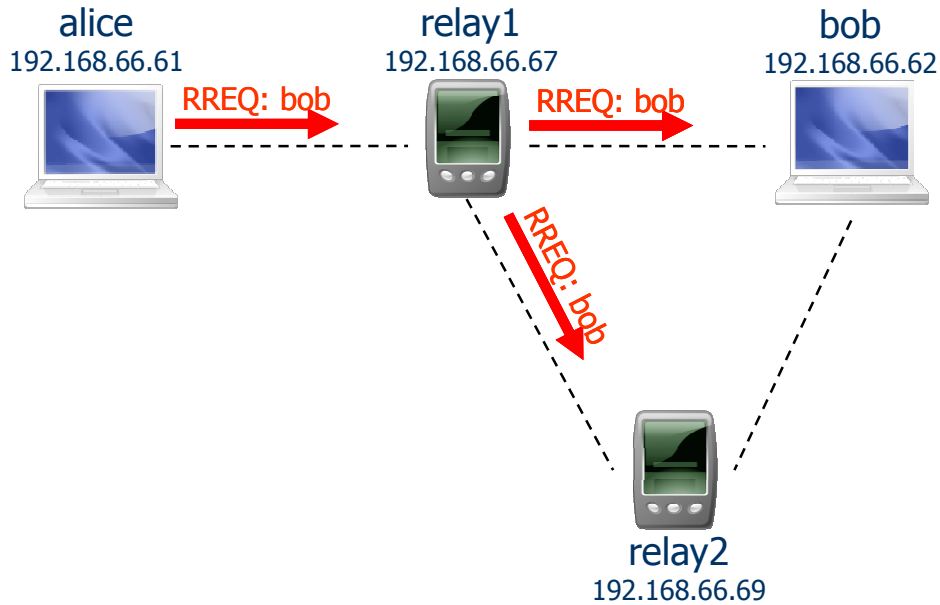
The A-SAODV adaptive secure routing protocol prototype

Davide Cerri, Alessandro Ghioni, Francesco Dolcini
CEFRIEL – Politecnico di Milano

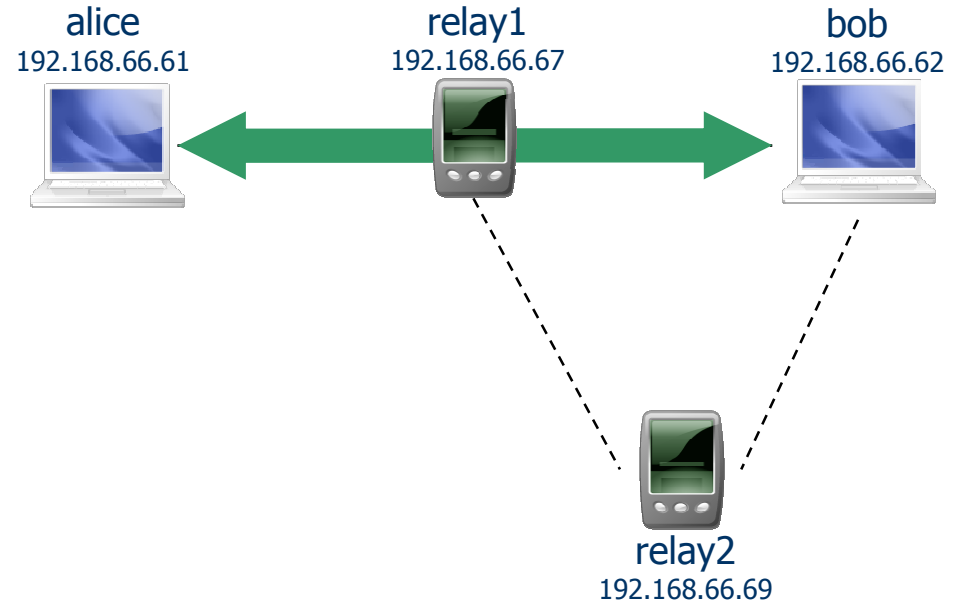
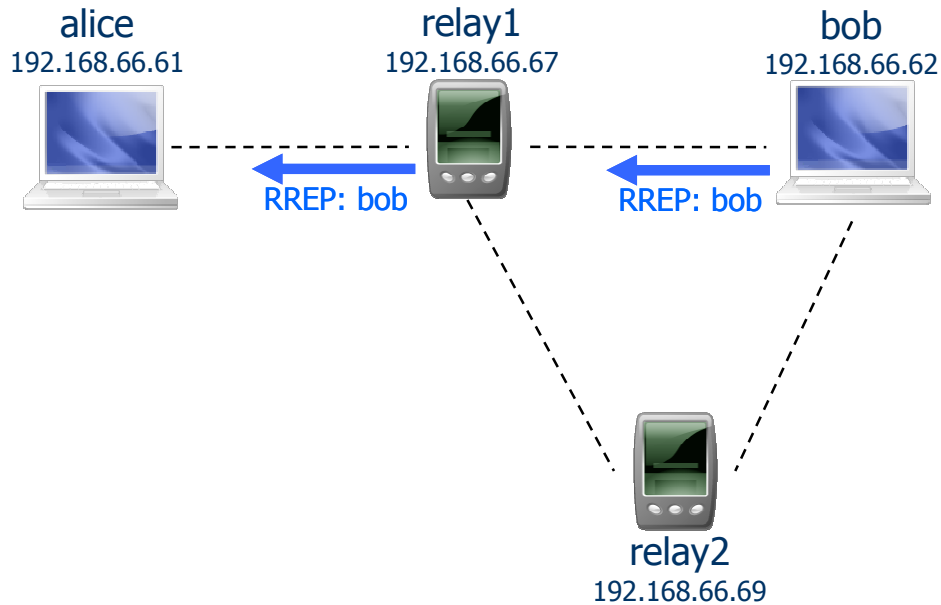
MobiHoc 2006

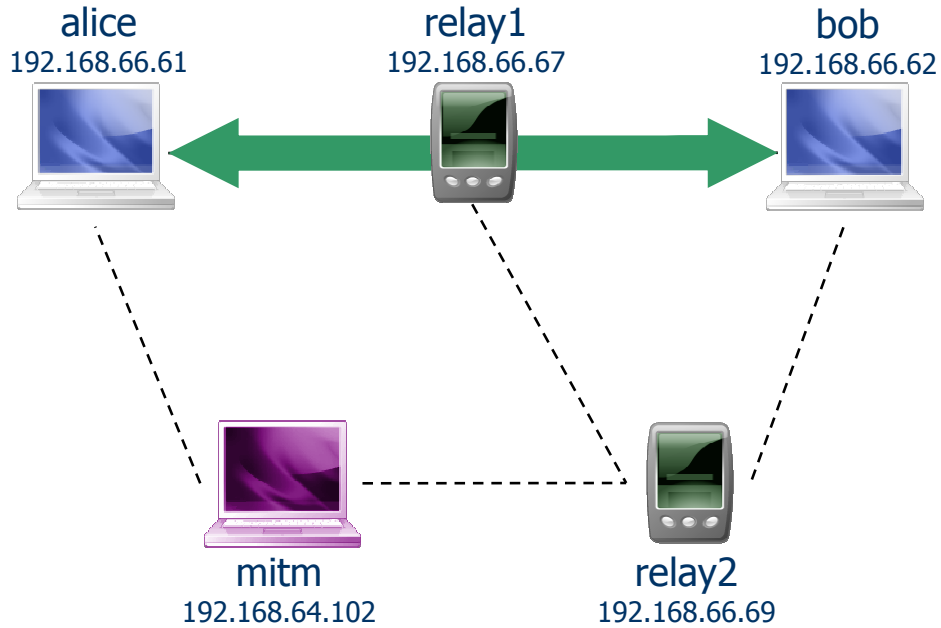
- **AODV** (Ad hoc On-demand Distance Vector): mobile ad hoc network routing protocol.
- Route discovery:
 - *Route Request* message (flooding)
 - establishes routes towards source node;
 - *Route Reply* message (unicast)
 - establishes routes towards destination node.
- **Collaboration:**
 - intermediate nodes may reply to a RREQ if they have a valid route towards the destination.

- AODV **does not take security into account**:
 - malicious nodes can **spoof and modify** routing messages;
- Many kinds of attack are possible, both **denial of service** and **man in the middle**:
 - with a DoS attack, the attacker may **prevent other nodes from communicating**;
 - with a MITM attack, the attacker puts himself “in the middle” of a communication between two nodes, and can **eavesdrop and modify data**.

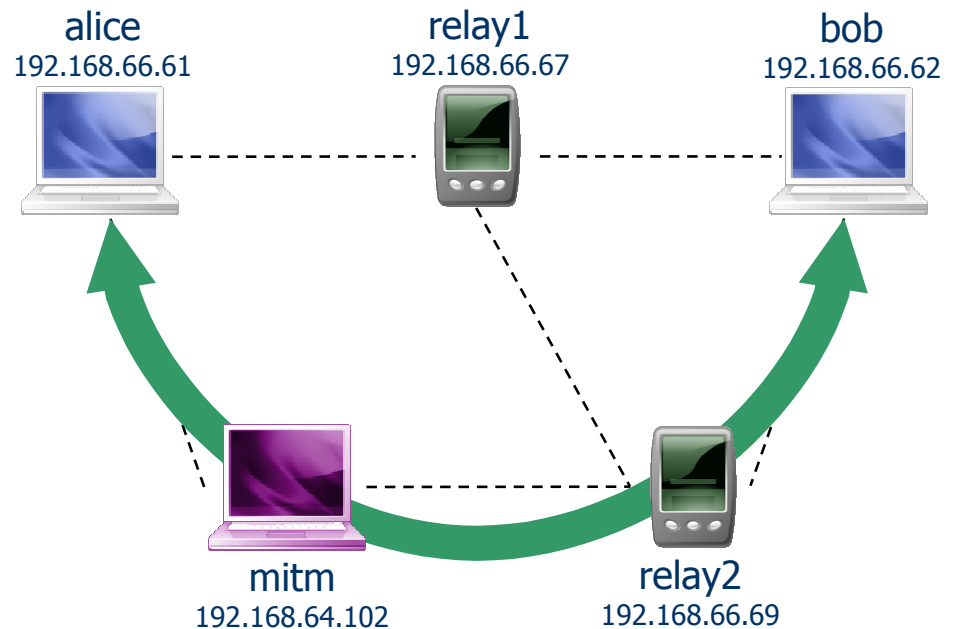
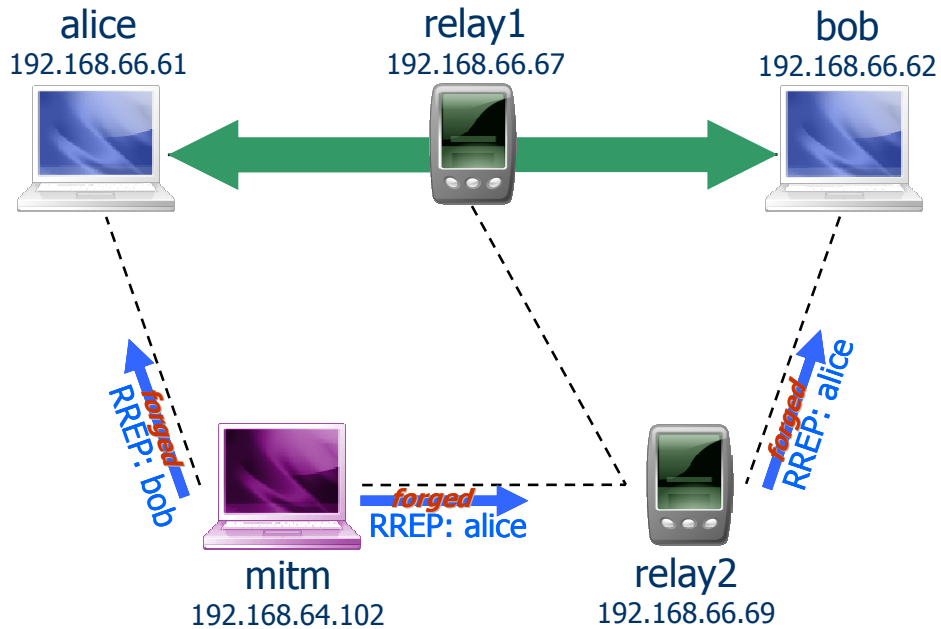


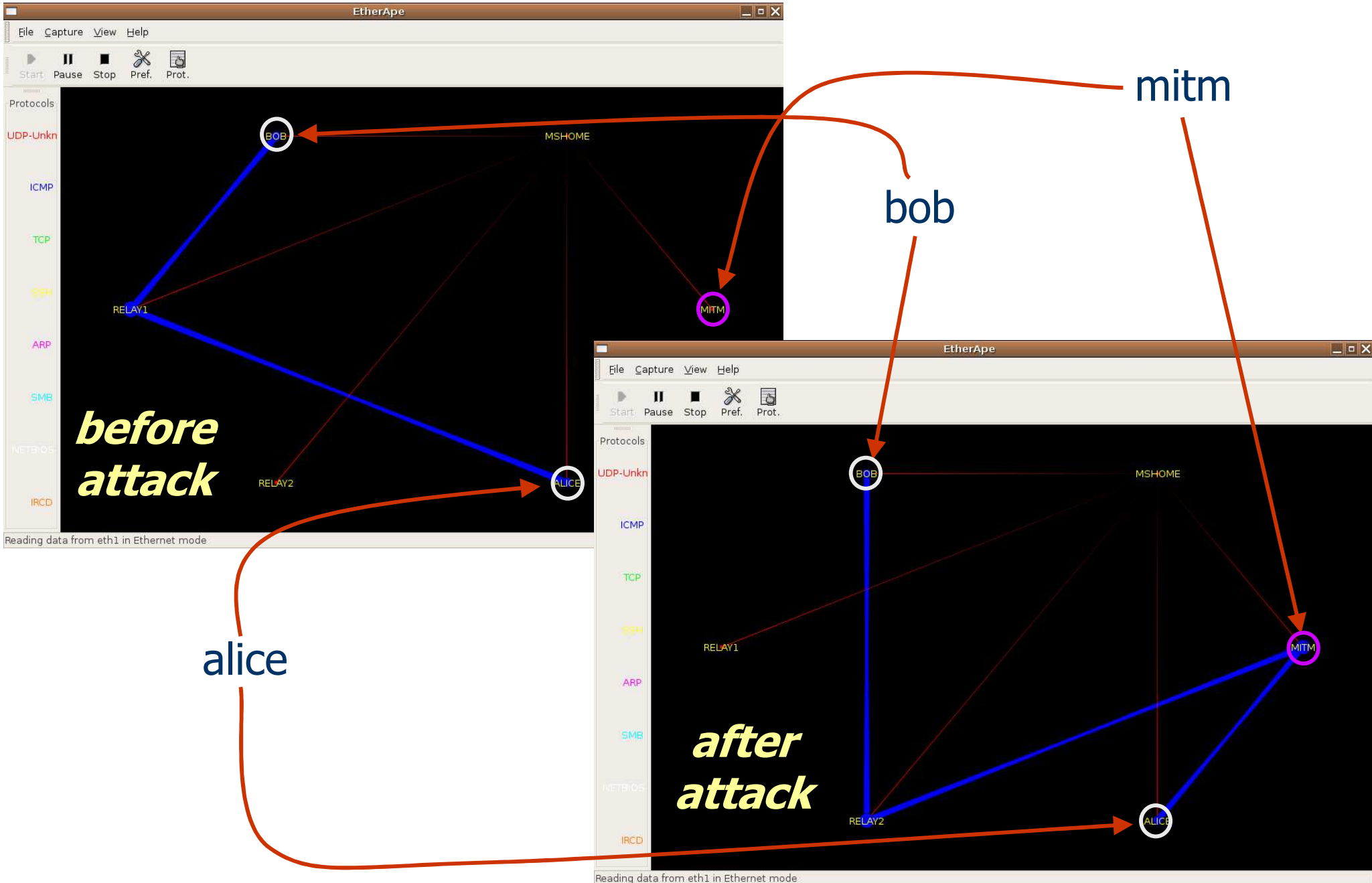
- Alice sends RREQ for Bob.
- Bob replies.
- A route is established between Alice and Bob; traffic flows through relay1.





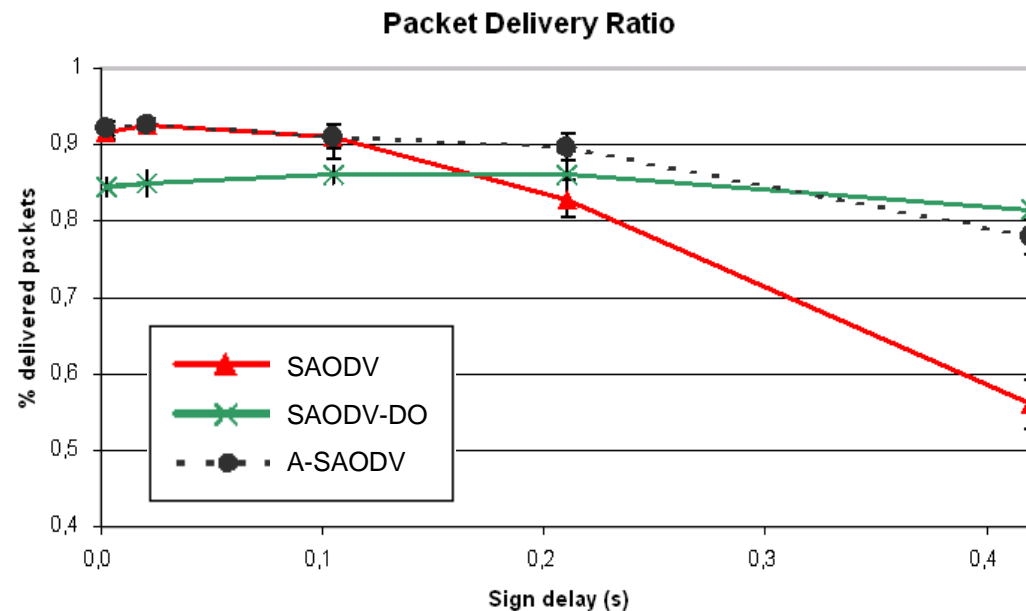
- The attacker arrives.
- The attacker sends forged RREPs to Alice and Bob.
- The traffic between Alice and Bob is rerouted through the attacker.

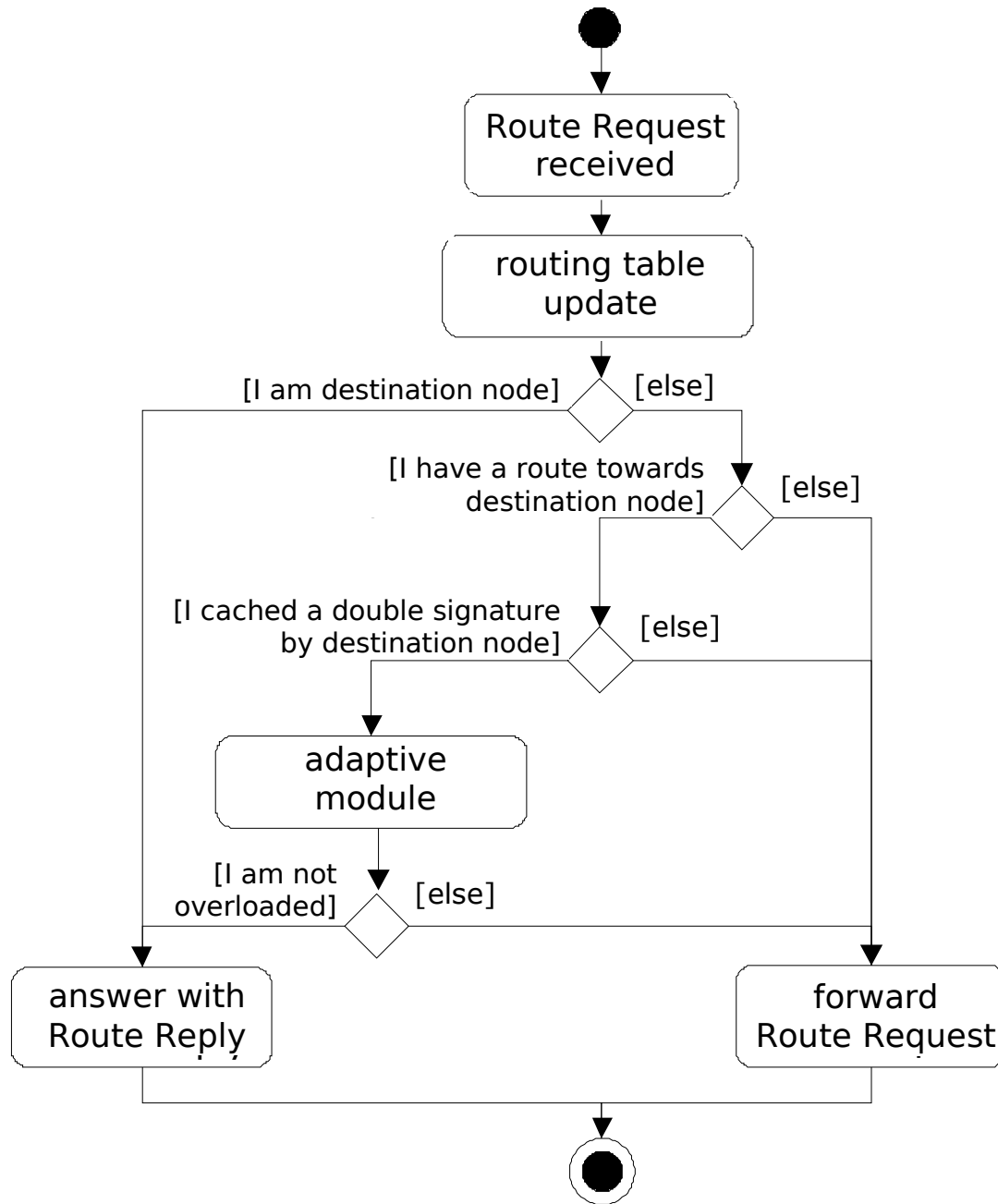




- **SAODV** (Secure AODV): proposal to add security to AODV.
 - Routing messages (RREQs, RREPs, RERRs) are signed.
 - Routing messages can contain a sort of “delegation” that allows intermediate nodes to reply on behalf of the destination node (“double signature”);
 - intermediate nodes must anyway also sign the reply with their own key.
- Double signature **preserves the AODV collaboration mechanism**, but:
 - in AODV this requires lightweight operations;
 - in SAODV collaboration is much heavier because of cryptographic signatures.

- What happens if we remove the double signature?
“SAODV-DO” (Destination Only).
 - “Uncollaborative” protocol: only the destination node is allowed to reply to a RREQ.
 - Performance worse than regular SAODV if signing time is low.
- **Our proposal: Adaptive SAODV (A-SAODV).**
 - Intermediate nodes reply to RREQs only if not overloaded.
 - Load estimation given by number of cryptographic operations enqueued.
 - Load estimation is compared with a threshold value that may be changed.

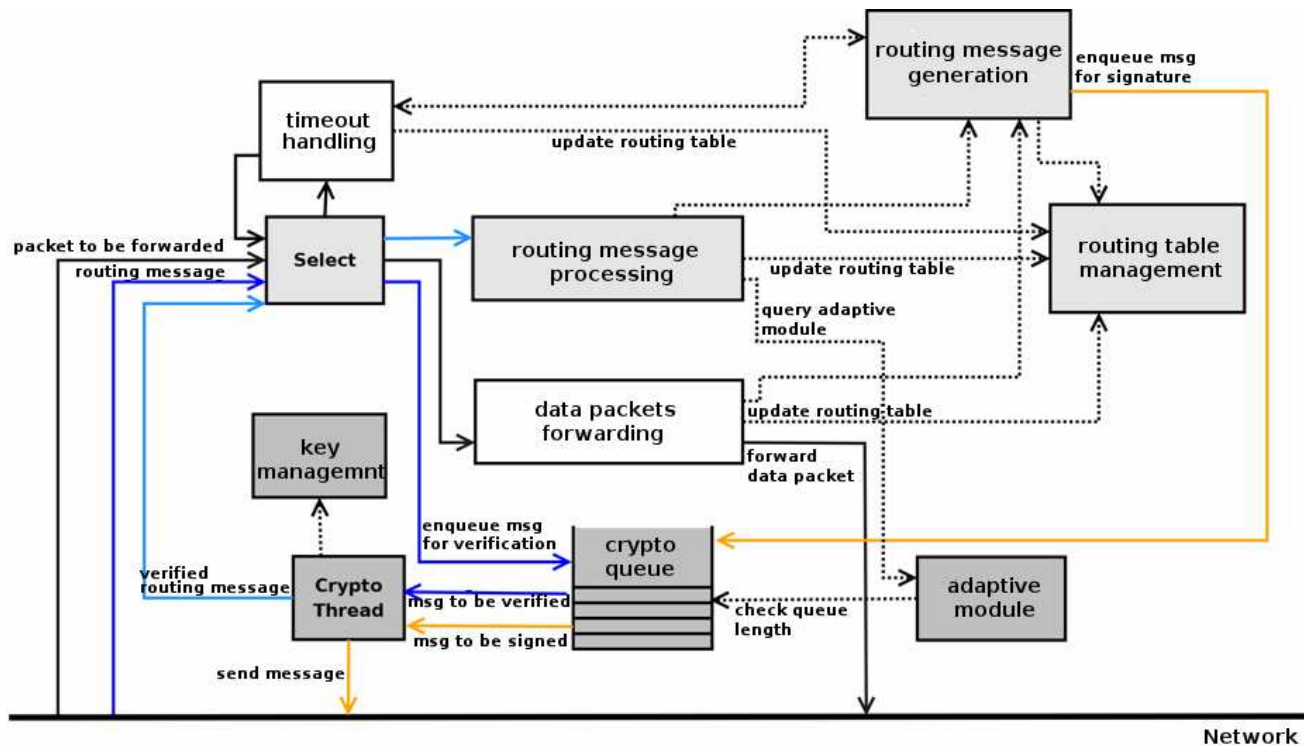




- If the intermediate node is able to reply (i.e., has the relevant information), it decides whether to do that in an adaptive way.
- The node decides by comparing the weight of its cryptographic queue with a threshold value:

$$\text{Weight} = N_{\text{sig}} \times \text{Weight}_{\text{sig}} + N_{\text{verif}} \times \text{Weight}_{\text{verif}}$$

- We **implemented SAODV** with this adaptive behaviour:
 - C language,
 - Linux operating system,
 - based on AODV-UU 0.8.1.
- We needed to move to a **multithreaded architecture** in order to manage cryptographic operations.

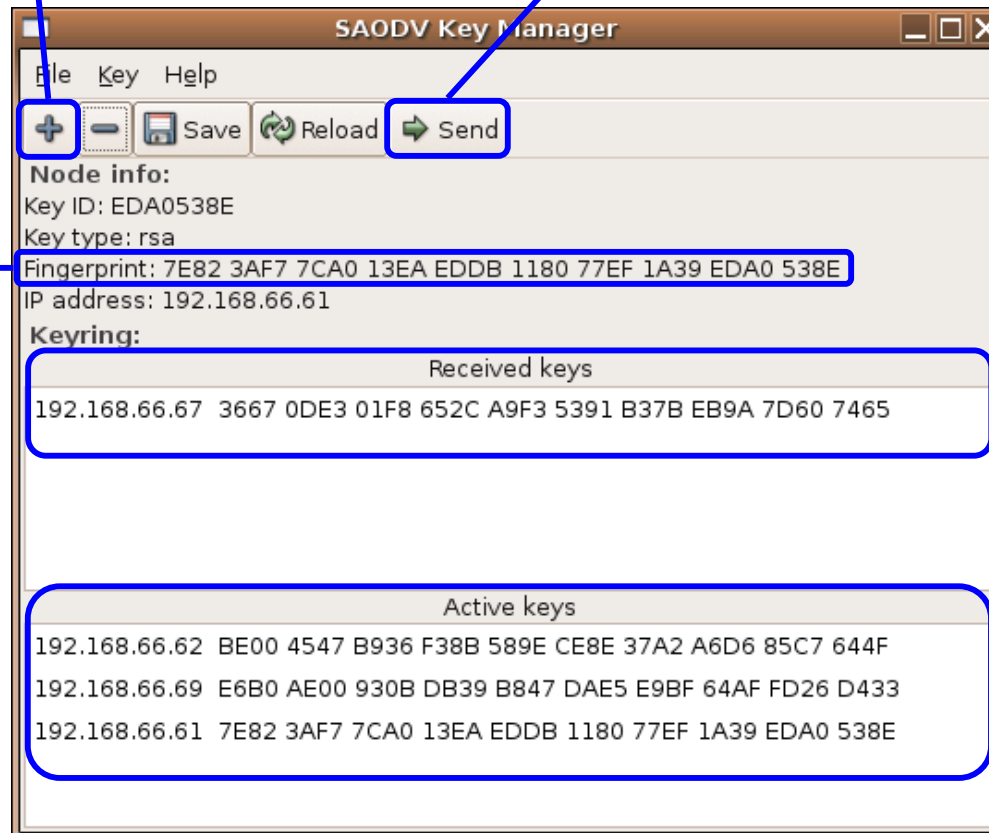


- We developed a minimal **graphical key manager** to be used with A-SAODV.
- Our key manager must be used in a bootstrap phase when all nodes are visible **within one hop**.
 - Routing operations cannot be performed before distributing keys (signatures would not be verified).
- Each user sends his/her key to others using **local broadcast** (255.255.255.255), then reads the key fingerprint aloud.
- Other users check if the fingerprint which is being read matches the one they received: if so, they **add the received key to their keyring**.
- This can be **extended with a web-of-trust** (PGP-like) mechanism in order to remove requirements of initial one-hop bootstrap and explicit approval of all keys by users.

accept key

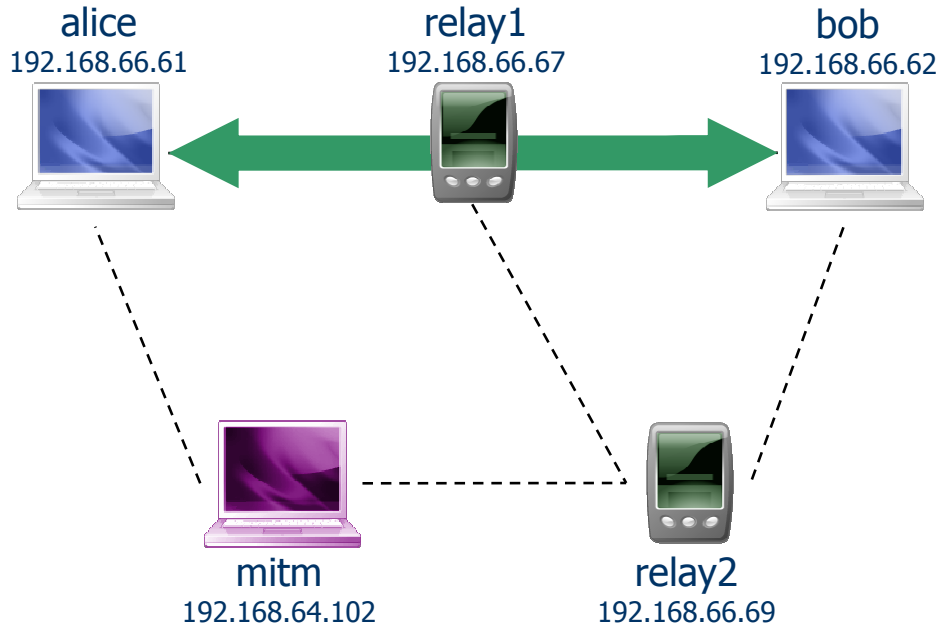
send own key

**own key
fingerprint**

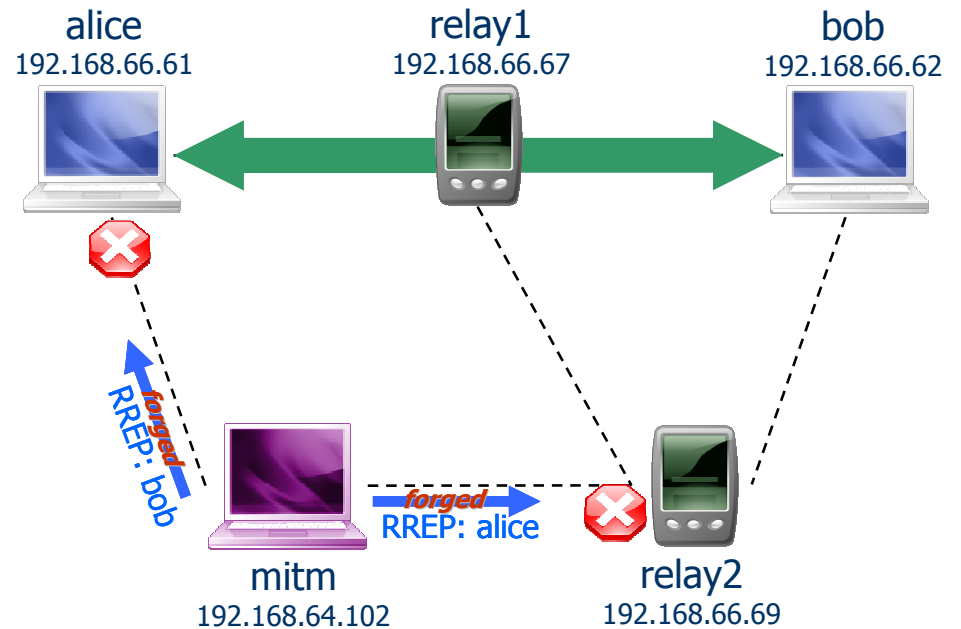
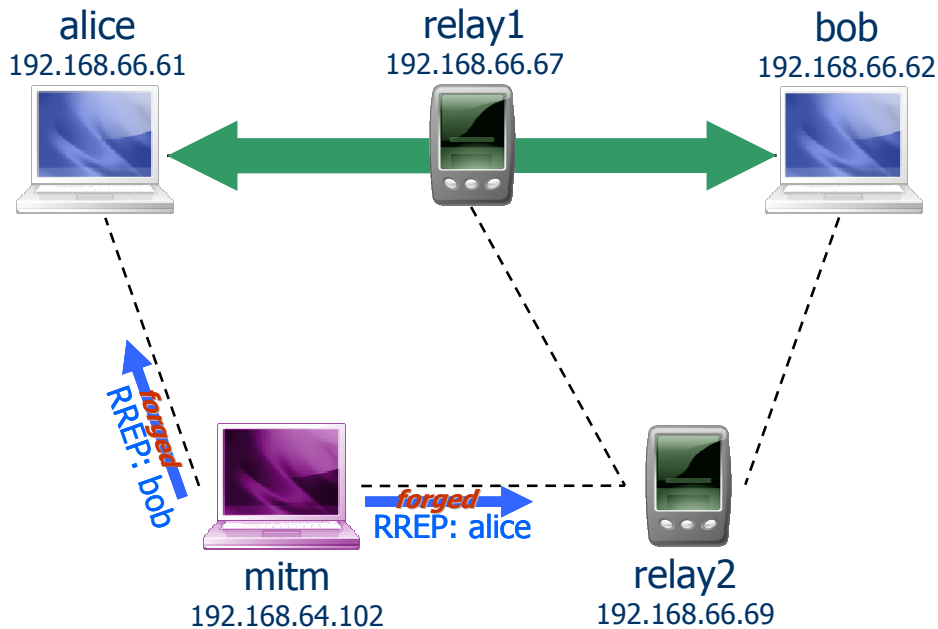


**received
keys**

**accepted
keys**



- The attacker arrives.
- The attacker sends forged RREPs to Alice and Bob.
- The attacker's neighbours recognize forged RREPs and discard them.



```

root@mwadhoc01: /opt/mais
root@mwadhoc01: /opt/mais
root@mwadhoc01: /opt/mais
11:41:25.733 crypto_queue_process: crypto_thread: verified aodv type:2 saodv type:65, result:FAILED!
11:41:25.733 crypto_queue_process: Verification took 0 millisc
11:41:25.733 crypto_queue_process: Verification (including queue traversal) took 0 millisc
11:41:25.789 rrep_sse_verify: failed rrep_sse signature verification, destination=192.168.66.63 dest_seqno=517
11:41:25.805 search_crypto_cache_node: found cached signature!
11:41:25.821 search_crypto_cache_node: found cached signature!
11:41:26.623 search_crypto_cache_node: found cached signature!
11:41:26.735 rrep_sse_verify: failed rrep_sse signature verification, destination=192.168.66.62 dest_seqno=10011
11:41:26.735 crypto_queue_process: crypto_thread: verified aodv type:2 saodv type:65, result:FAILED!
11:41:26.735 crypto_queue_process: Verification took 0 millisc
11:41:26.735 crypto_queue_process: Verification (including queue traversal) took 0 millisc
11:41:26.813 search_crypto_cache_node: found cached signature!
11:41:26.828 rrep_sse_verify: failed rrep_sse signature verification, destination=192.168.66.63 dest_seqno=517
11:41:26.860 search_crypto_cache_node: found cached signature!
11:41:27.633 search_crypto_cache_node: found cached signature!
11:41:27.737 rrep_sse_verify: failed rrep_sse signature verification, destination=192.168.66.62 dest_se
11:41:27.737 crypto_queue_process: crypto_thread: verified aodv type:2 saodv type:65, result:FAILED!
11:41:27.738 crypto_queue_process: Verification took 0 millisc
11:41:27.738 crypto_queue_process: Verification (including queue traversal) took 0 millisc
11:41:27.853 search_crypto_cache_node: found cached signature!
11:41:27.869 search_crypto_cache_node: found cached signature!
11:41:27.877 search_crypto_cache_node: Number of messages:1180, hit rate:64.07
11:41:27.877 rrep_sse_verify: failed rrep_sse signature verification, destination=192.168.66.63 dest_se
11:41:28.653 search_crypto_cache_node: found cached signature!
11:41:28.738 rrep_sse_verify: failed rrep_sse signature verification, destination=192.168.66.62 dest_se
11:41:28.739 crypto_queue_process: crypto_thread: verified aodv type:2 saodv type:65, result:FAILED!
11:41:28.739 crypto_queue_process: Verification took 0 millisc
11:41:28.739 crypto_queue_process: Verification (including queue traversal) took 1 millisc
11:41:28.875 search_crypto_cache_node: found cached signature!
11:41:28.891 search_crypto_cache_node: found cached signature!
11:41:28.892 rrep_sse_verify: failed rrep_sse signature verification, destination=192.168.66.63 dest_seqno=517
11:41:29.663 search_crypto_cache_node: found cached signature!
11:41:29.740 rrep_sse_verify: failed rrep_sse signature verification, destination=192.168.66.62 dest_seqno=10014
11:41:29.740 crypto_queue_process: crypto_thread: verified aodv type:2 saodv type:65, result:FAILED!
11:41:29.741 crypto_queue_process: Verification took 0 millisc
11:41:29.741 crypto_queue_process: Verification (including queue traversal) took 0 millisc
11:41:29.877 search_crypto_cache_node: found cached signature!
11:41:29.929 rrep_sse_verify: failed rrep_sse signature verification, destination=192.168.66.63 dest_seqno=517

```

The attacker cannot forge valid signatures coming from Bob, so Alice discards the attacker's routing messages.

Davide Cerri, Alessandro Ghioni

CEFRUEL – Politecnico di Milano

{cerri, ghioni}@cefriel.it

<http://saodv.cefriel.it/>

