Validation of routing security for mobile ad hoc networks

Ana Cavalli and Jean-Marie Orset

Groupe des Ecoles des Télécommunications/Institut National des Télécommunications

Outline

- Ad Hoc Networks validation
- Attack Detection Systems
- Approach based on monitoring techniques
- Approach by Invariants
- Open problems

Ad Hoc Network validation

Ad hoc Networks Validation

- **Validation research work is mainly focused on the routing security (ad hoc and sensor networks) + works on the integrity of messages and the authentification of nodes**

- **How to use formal techniques for verification and testing to cover these aspects ?**
  - Formal modeling of routing protocols (to facilitate verification and test by providing a formal model)
  - Verification. To insure the correction of the routing protocol specification (liveness, absence of loops, etc.) and their security mechanisms (messages integrity, authentification, attacks prevention)
  - Test (to identify failures in the implantation, for instance: to detect attacks on the routing process)

Security mechanisms in ad hoc networks

- **Cryptography to insure integrity and authentification**
  - Prevention but not detection
  - Diminish the number of attacks without eliminating them
  - Protection with respect some types of attacks
  - It doesn’t allow to detect and treat malicious nodes
  - It doesn’t permit to identify attacks (it allows one mutual identification of nodes without detection of all attacks, for instance denial of service)
  - Uses mechanisms very heavy based on strong hypothesis (presence of a public key infrastructure and synchronization of nodes)
- **Failure/limitations of existing solutions**

  => need of mechanisms for attacks detection

Attack Detection Systems

Attacks Detection Systems

Traditionnally of 2 types

- **Approach by signature**
  - Based on the analysis of the information exchanged by the nodes looking for attacks that correspond to known patterns (*pattern matching*). Detection of behaviors that are close to the signature of a known attack
  - Example: “Network grep” – look for the string of characters in network connection that could indicate an attack is in progress
- **Behavior approach**
  - Detection of behaviors that are not close to the normal behavior of the node
  - Application of statistics measures or heuristics to subsequent events in order to determine if they conform to the « normal » model/statistics
  - If the events do not follow a « normal » probability then it is necessary to generate an alarm

Attacks detection systems

Limitations of the signature approach:

- It only permits to detect known attacks
- Difficult to maintain updated signatures
- Absence of a centralized entity to supervise the traffic
- Can be *dupée* (attacks that slightly vary the signature)

Limitations of the behaviour approach:

- Not clear distinction between normal or abnormal behavior
- Need to process a big amount of data
- Reduced efficiency
- Too many false positives

Vulnerabilities of ad hoc networks

- **Absence of infrastructure**
  - Router, DNS server, certification authority
- **Wireless transmission medium**
- **Strong mobility**
- **Limited capabilities**
  - Band width, autonomy, processing power

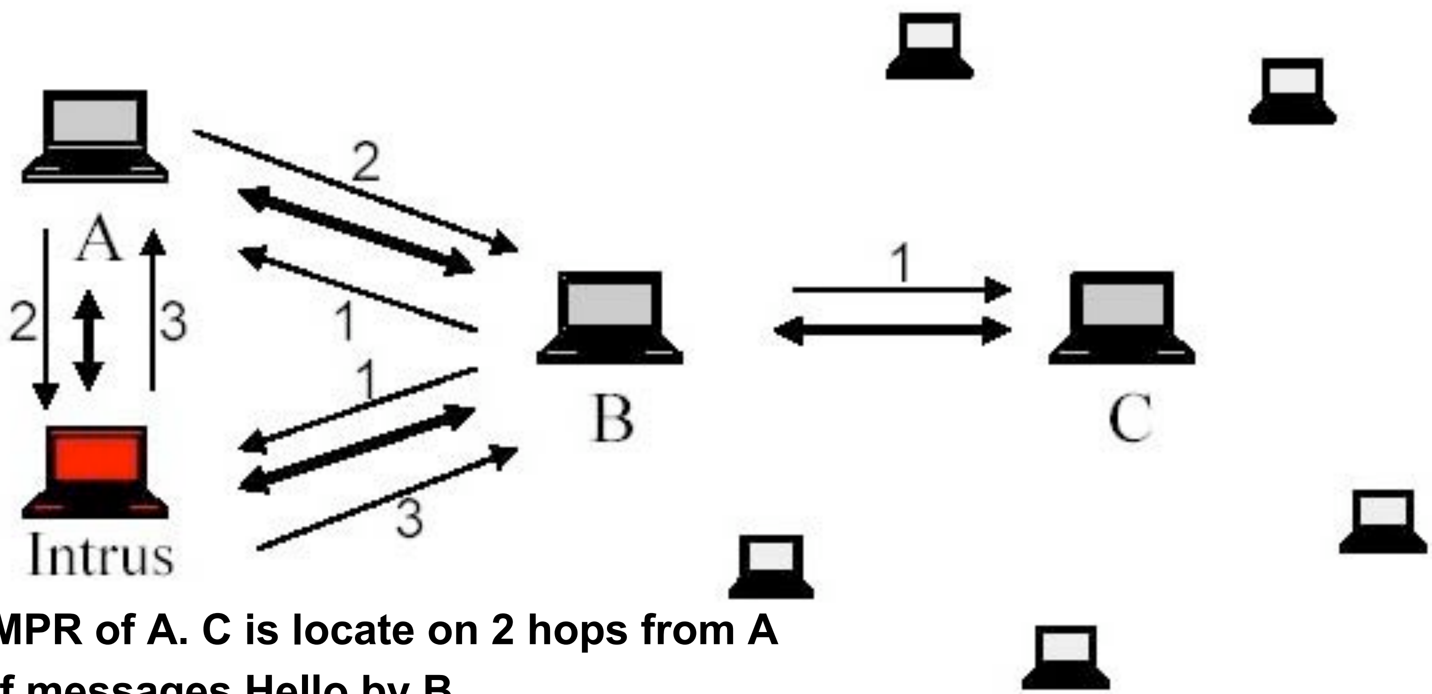
Types of attacks

- **Current attacks in mobile wireless networks:**
  - Sniffing (data, localization, etc..)
  - Identity usurpation (Spoofing IP, ARP, etc…)
  - Modification
  - Insertion => creation of loops
  - Denial of service (DoS)

Types of attacks

- **Attacks characteristics of Ad Hoc routing:**
  - Non cooperation (Selfishness)
  - Creation of a tunnel or private connections (Wormhole)

Example of an attack to OLSR



B is the MPR of A. C is locate on 2 hops from A

1. Sent of messages Hello by B

2. Sent of messages Hello by A

3. Insertion of a message Hello by the Intruder announcing to A, B, C a symmetric link

Consequences :

- **Selection of the Intruder as a MPR by A**
- **The traffic of A towards C pass through the Intruder**

Approach based on monitoring techniques

Approach based on monitoring techniques

- **Exhaustive functional analysis**

  To compare the input/output traces (messages sent and received) with the specification given as an EFSM (to facilitate the detection of abnormal behaviors).

- **Analysis by invariants**

  To check invariants that describe security properties on the traces in order to detect behaviors that violate them. The invariants are given under the form of a logical formula (temporal-deontic formula).

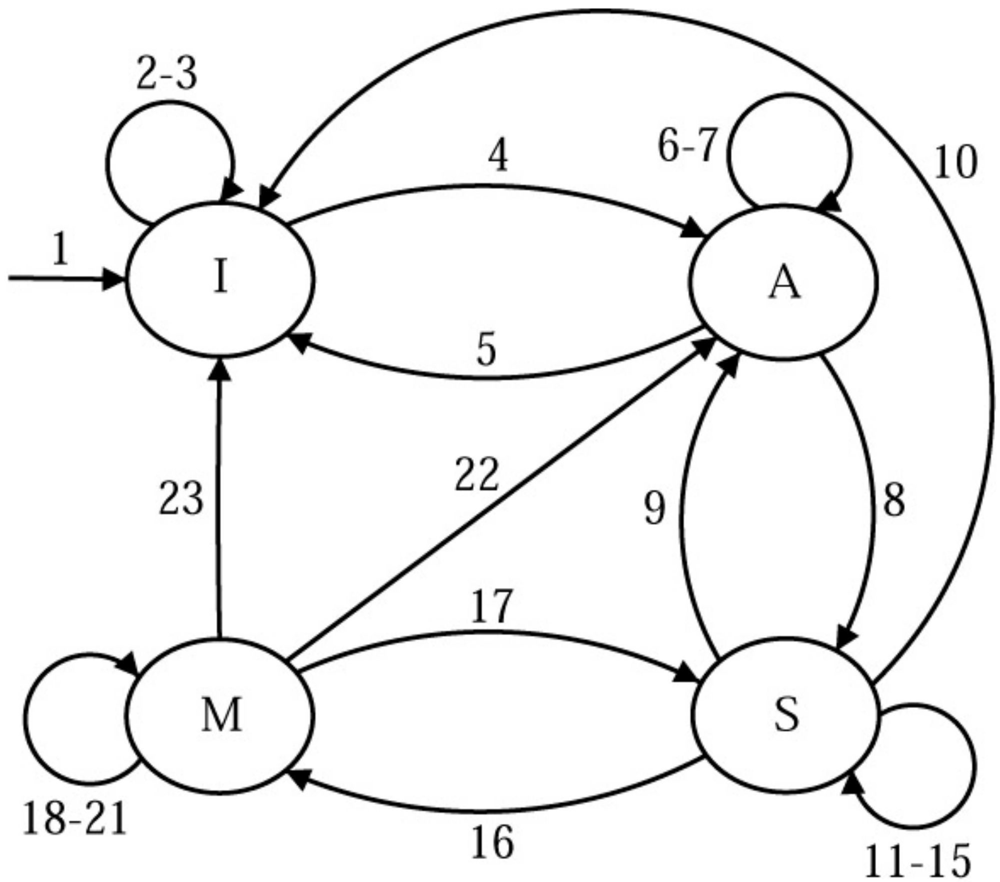
Extended Finite State Machines

- **Specification of OLSR as a EFSM (Extended Finite State Machine)**

- **The EFSM (Extended Finite State Machine) are characterized by :**
  - I/O events with or without parameters
  - a predicate to be satisfied
  - actions to be performed

- **Execution traces (sequence of I/O couples) of the system under test.**

OLSR EFSM

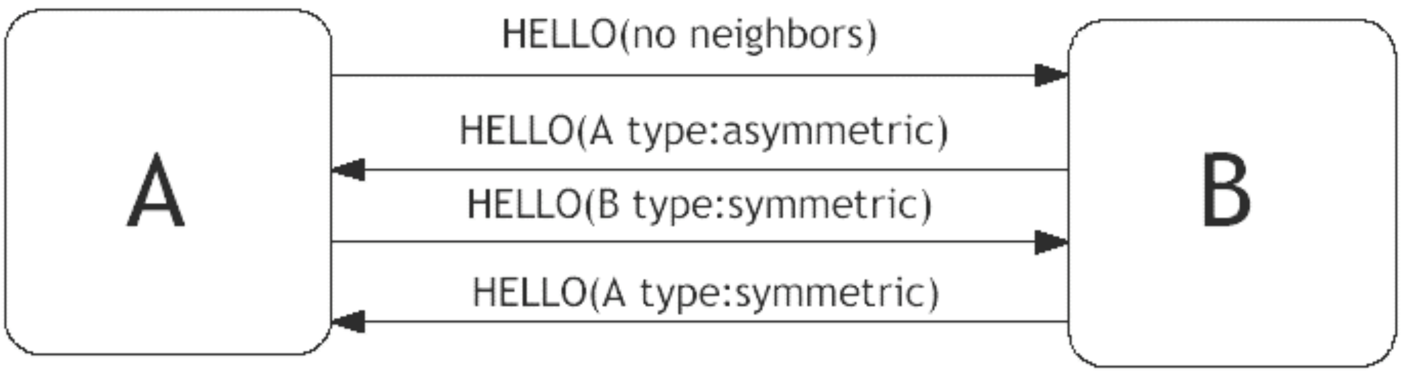


OLSR EFSM obtained from RFC 3626

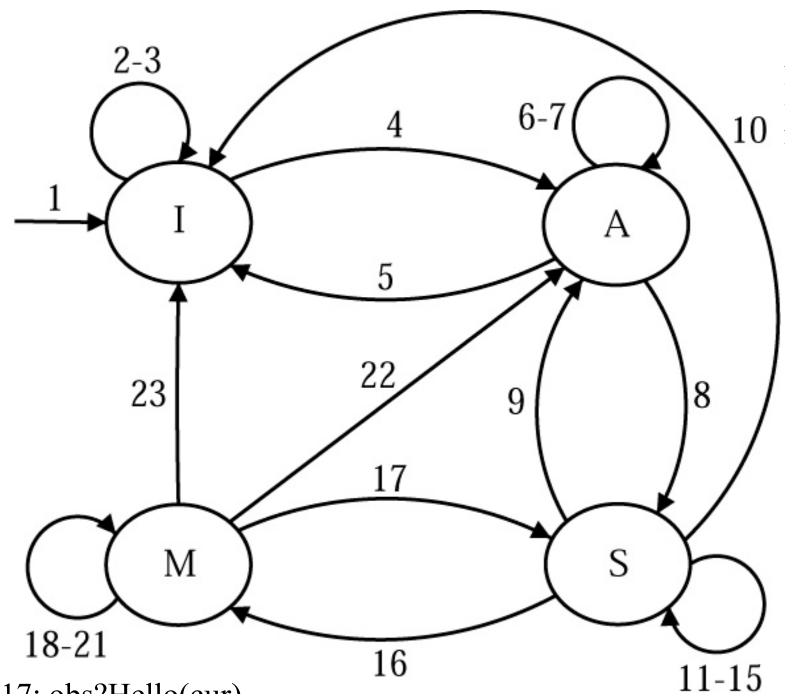
Neighbor discovery

- Periodical emission of the packets *hello*
  - – The *hello* include the list of nodes detected (*entendus*) and the type of link

=> The nodes know their neighbors and those two hops away



Example



10: UpdateTimerOut
A: reset UpdateTimer;
Remove(obs,AsymList);
reset SentHello; remove(obs,MprList)\\

8: obs?Hello(cur)
P: (cur=ASYM AND SentHello=true)
OR (cur=SYM AND obs$\in$AsymList)\\
A: reset UpdateTimer

11: HelloTimerOut / cur!Hello(obs)
A: set obs=SYM; reset HelloTimer; remove(obs,MprList)

12: HelloTimerOut / cur!Hello(obs)
A: set obs=MPR; reset HelloTimer; add(obs,MprList)

13: cur!Data()
P: obs$\in$MprList

14: obs?TC(cur)
P: cur=MPRSEL AND obs$\in$MprList

15: obs?Hello(cur)
P: cur=SYM OR cur=ASYM

17: obs?Hello(cur)
P: cur=SYM OR cur=ASYM
A: reset UpdateTimer; Remove(obs,MprSelList)

16: obs?Hello(cur)\\
P: cur=MPR\\
A: Add(obs,MprSelList);
reset UpdateTimer; reset TcTimer

Detection process

- **The process of verification/detection consist of comparing the I/O traces (messages sent and received) with the specification**
  - The trace needs to be accepted as a word of the EFSM
- **The checking is performed by the application of an algorithm (backward checking) previously defined**

Example

Insertion of fault messages 'Hello'

- **One possible trace is:**

  (start)

  HelloTimerOut / cur!Hello()

  UpdateTimerOut

  obs?Hello(cur) / cur=SYM

  cur: courant node

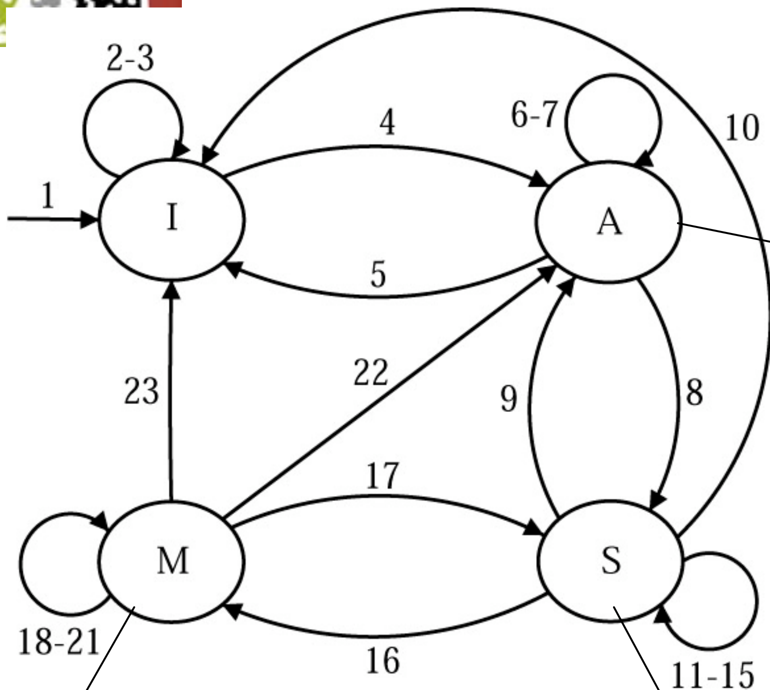
  obs: observed node

  (an intruder announce a non existing symmetric link to their neigbourghs)

I. 1) Starting from the last event: looking for the corresponding transitions (8,15,17)

Example

=> correspondance with transitions n° 8,15 &17



10: UpdateTimerOut
A: reset UpdateTimer;
Remove(obs,AsymList);
reset SentHello; remove(obs,MprList)

8: obs?Hello(cur)
P: (cur=ASYM AND SentHello=true)
OR (cur=SYM AND obs€AsymList)
A: reset UpdateTimer

11: HelloTimerOut / cur!Hello(obs)
A: set obs=SYM; reset HelloTimer; remove(obs,MprList)

12: HelloTimerOut / cur!Hello(obs)
A: set obs=MPR; reset HelloTimer; add(obs,MprList)

13: cur!Data()
P: obs€MprList

14: obs?TC(cur)
P: cur=MPRSEL AND obs€MprList

17: obs?Hello(cur)
P: cur=SYM OR cur=ASYM
A: reset UpdateTimer;
Remove(obs,MprSelList)

16: obs?Hello(cur)
P: cur=MPR
A: Add(obs,MprSelList); reset TcTimer

15: obs?Hello(cur)
P: cur=SYM OR cur=ASYM
A: reset UpdateTimer

Example

I.2) Looking for possible previous configurations:

State: A; Parameters: cur = SYM, obs∈AsymList

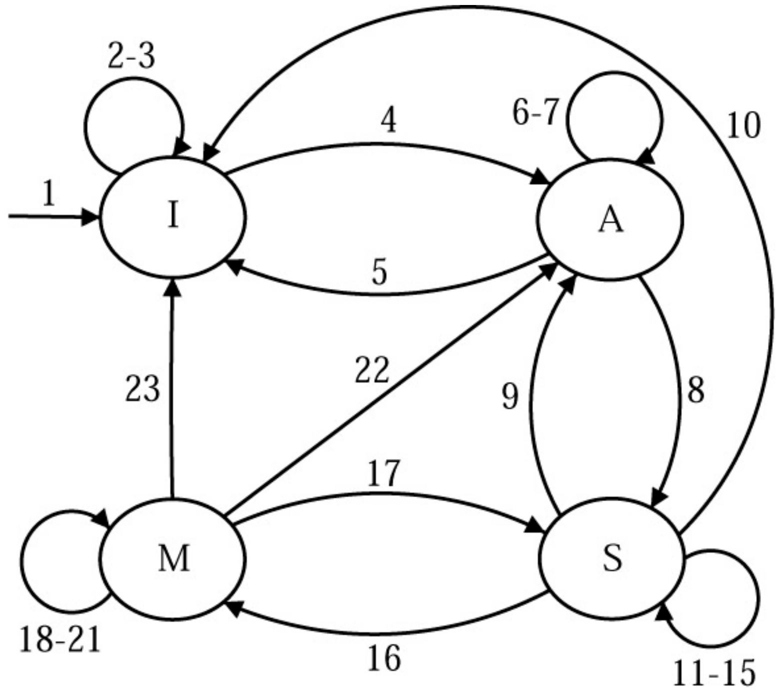
State: S; Parameters: cur = SYM

State: M; Parameters: cur = SYM, obs∈MprList

II.1) We restart the process on the precedent transition (UpdateTimer)

Example

=> No one corresponds!



10: UpdateTimerOut
A: reset UpdateTimer;
Remove(obs,AsymList);
reset SentHello; remove(obs,MprList)

8: obs?Hello(cur)
P: (cur=ASYM AND SentHello=true)
OR (cur=SYM AND obs€AsymList)

11: HelloTimerOut / cur!Hello(obs)
A: set obs=SYM; reset HelloTimer; remove(obs,MprList)

12: HelloTimerOut / cur!Hello(obs)
A: set obs=MPR; reset HelloTimer; add(obs,MprList)

13: cur!Data()
P: obs€MprList

14: obs?TC(cur)
P: cur=MPRSEL AND obs€MprList

15: obs?Hello(cur)
P: cur=SYM OR cur=ASYM

17: obs?Hello(cur)
P: cur=SYM OR cur=ASYM
A: Remove(obs,MprSelList)

16: obs?Hello(cur)
P: cur=MPR
A: Add(obs,MprSelList);
reset TcTimer

Example

II.2) There is not a transition that satisfies the constraints

(the transitions with the event *UpdateTimerOut* do not go to states A, S or M)

=> Violation of the specification by a transfer error !!

Discussion

- No false positives
- Exhaustive approach
- No errors identification
  - Conformance errors / security failures?
- It doesn’t allow detection of attacks that don't violate the specification (for instance DoS)

Approach by Invariants

Approach by Invariants

- Extraction of the RFC 3626 relevant properties and identification of the basics security properties

- Transformation of these properties under the form of an invariant using a language combining deontic and temporal logic

Syntax of the language

- **Operators of modal logic: ⊕ & ⊖**
- **Deontic modalities: $\mathcal{F}$, $\mathcal{O}$ (i.e. resp. Forbidden, Obligatory)**
- **Operators to indicate an action in a formula:**
  – done (α) & start (β)
- **Operators of classical logic…**

=> with this language we can describe complex properties

* Based on logic language defined by Nora and Frédéric Cuppens, ENST Bretagne

Example of invariant

- **Links verification**
  - $\mathcal{F}\Big(done(n?M, Hello(n : Asym)) | \neg \ominus \big(done(n!Hello())\big)\Big)$
  - $\mathcal{F}\Big(done(n?M, Hello(n : Sym)) | \neg \ominus \big(done(n!Hello(M : Asym))\big)\Big)$
  - $\mathcal{F}\Big(done(n?M, Hello(n : MPR)) | \neg \ominus \big(done(n!Hello(M : Sym))\big)\Big)$

Discussion

- **It allows efficiently detecting the attacks during the establishment of links (that are the most important attacks on pro-active protocols)**
- **Secure the establishment of links mechanisms**
  - The nodes can detect if there are false neighbors
- **The approach can be optimized**
  - The monitoring can be limited to nodes that are MPR

Open Problems

Open problems

- **Open problems**
  - Some properties can be verified by local observations, others need global ones (for instance, interoperability)

- **How to identify and eliminated malicious nodes ?**
  - Once the attack is detected, what action needs to be taken? The suspicious node is excluded? It is denounced to the neighbors ?
  - How to avoid that a suspicious node announce inexistent attacks provoking the exclusion of normal nodes?

Work in progress

- **Verification of the consistance of invariants**

- **Correlation of traces of different nodes (in order to detect distributed attacks)**

- **Analysis of traces in order to detect the state in which the implementation is**
  - The identification of the initial state could permit executing a property (described, for instance, as a finite state machine) on the traces
  - Useful for supervision

- **On line monitoring**