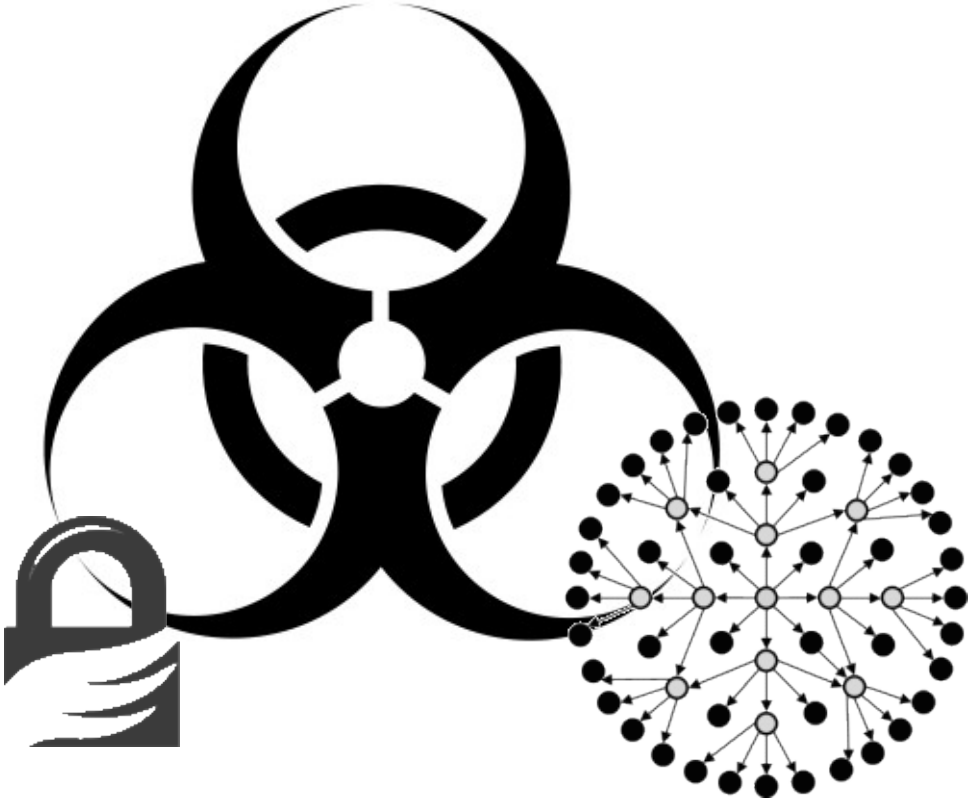
The biohazard of coupling PGP with OLSR

the (sad story of the) web of trust plug-in and its defects

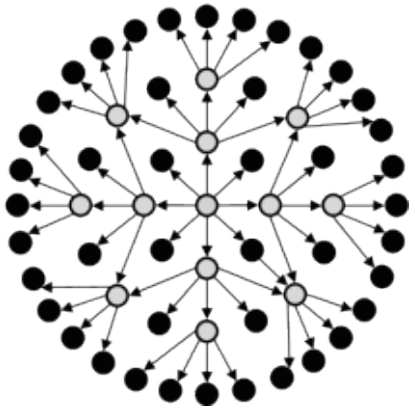


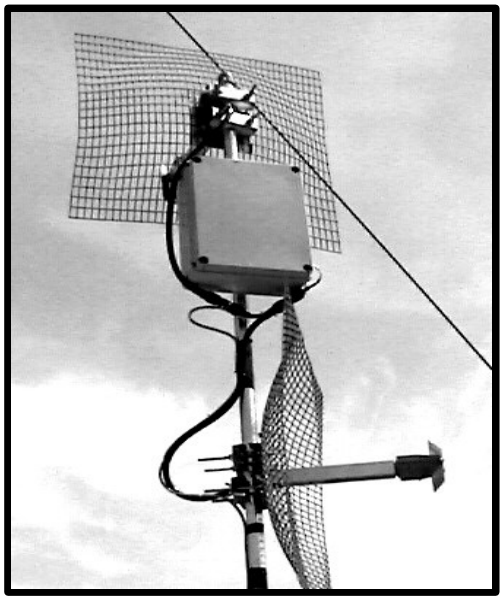
clauz and ZioPRoTo

ninux.org

OLSR

0		31
Packet Length		Packet Sequence Number
Message Type	Vtime	Message Size
Originator Address		
Time To Live	Hop Count	Message Sequence Number
MESSAGE		
Message Type	Vtime	Message Size
Originator Address		
Time To Live	Hop Count	Message Sequence Number
MESSAGE		
⋮		





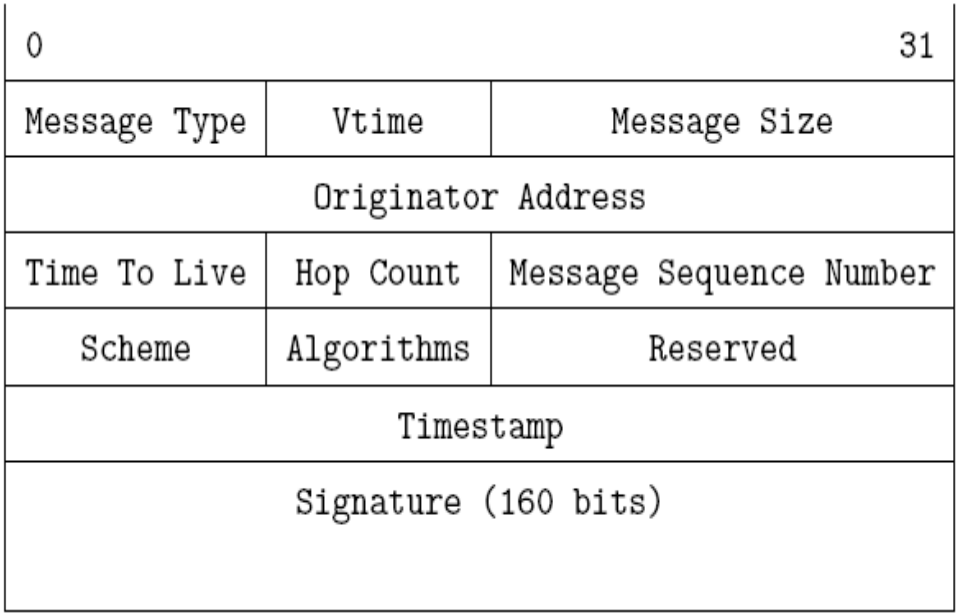
PGP

- Mixed Asymmetric/Symmetric cryptography for encryption and signing
- Decentralized web of trust
- Used prevalently for e-mails
- Social:
  - Zimmerman's motivations for writing PGP
  - Key signing parties

Biohazard

- PGP-**sign** packets hop by hop
  - Assumption: *I trust what my trusted neighbors trust*
- Use the trust associated with PGP signatures to build **multiple routing tables**, each with a different level of trust
- Then use these routing tables to implement **trusted routes**

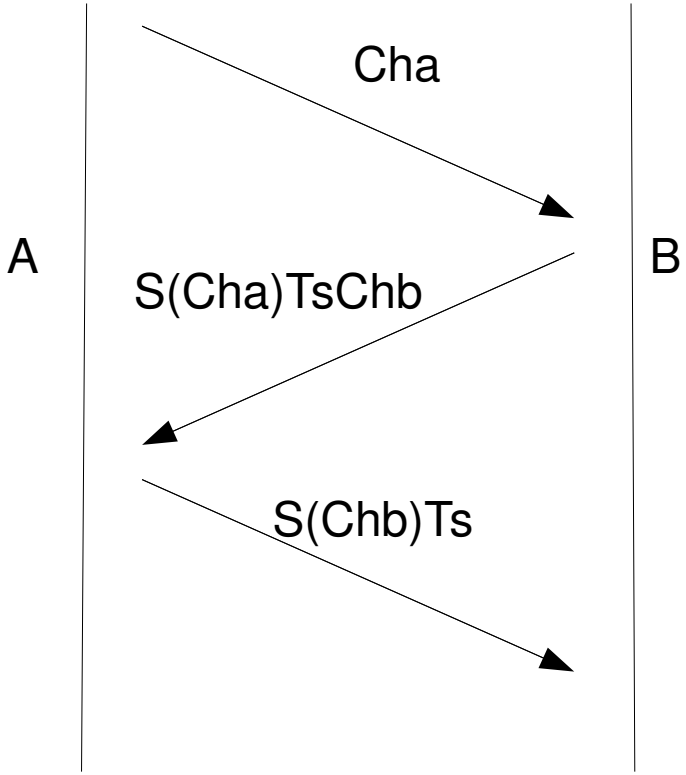
Secure OLSR plug-in



0		31
Message Type	Vtime	Message Size
Originator Address		
Time To Live	Hop Count	Message Sequence Number
Scheme	Algorithms	Reserved
Timestamp		
Signature (160 bits)		

- By Tønnesen (et al.)
- Nodes share a common secret
- OLSR message to sign OLSR packets

Secure OLSR plug-in - timestamp exchange

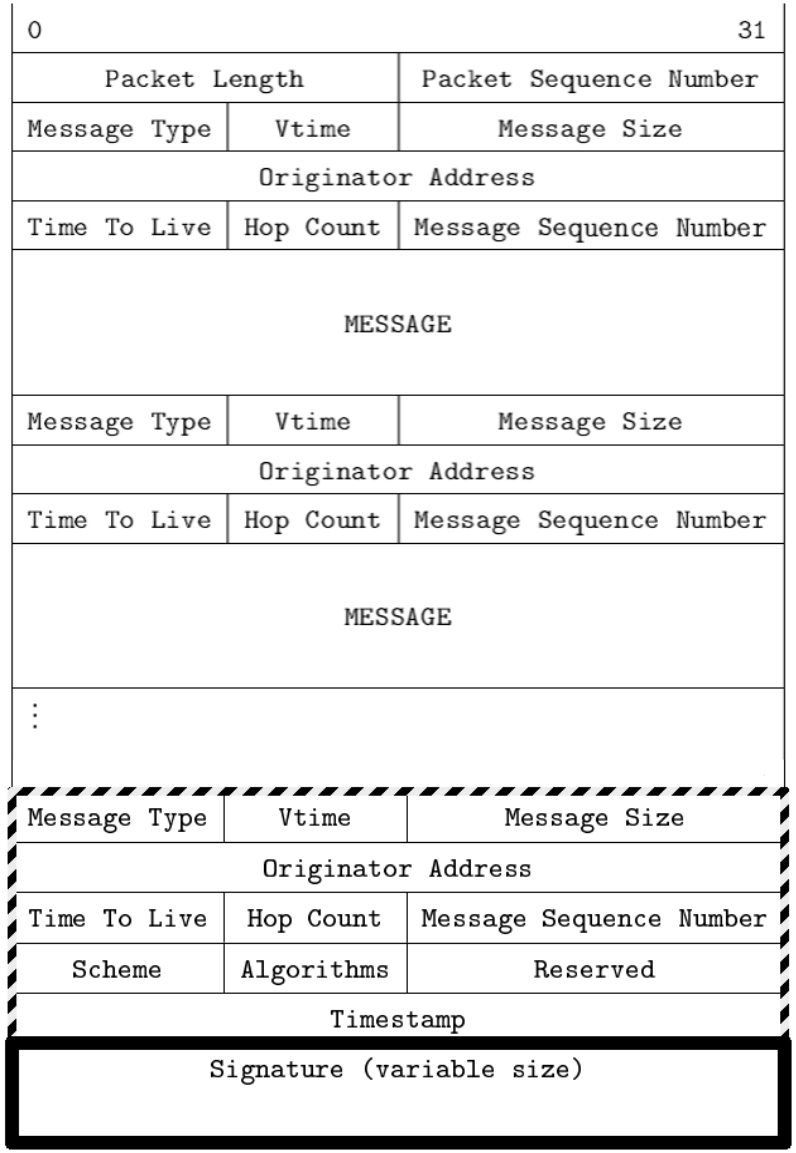


0		31
Message Type	Vtime	Message Size
Originator Address		
Time To Live	Hop Count	Message Sequence Number
Destination		
Random value "challenge"		
Signature (160 bits)		

0		31
Message Type	Vtime	Message Size
Originator Address		
Time To Live	Hop Count	Message Sequence Number
Destination		
Random value "challenge"		
Timestamp		
Response Signature (160 bits)		
Signature (160 bits)		

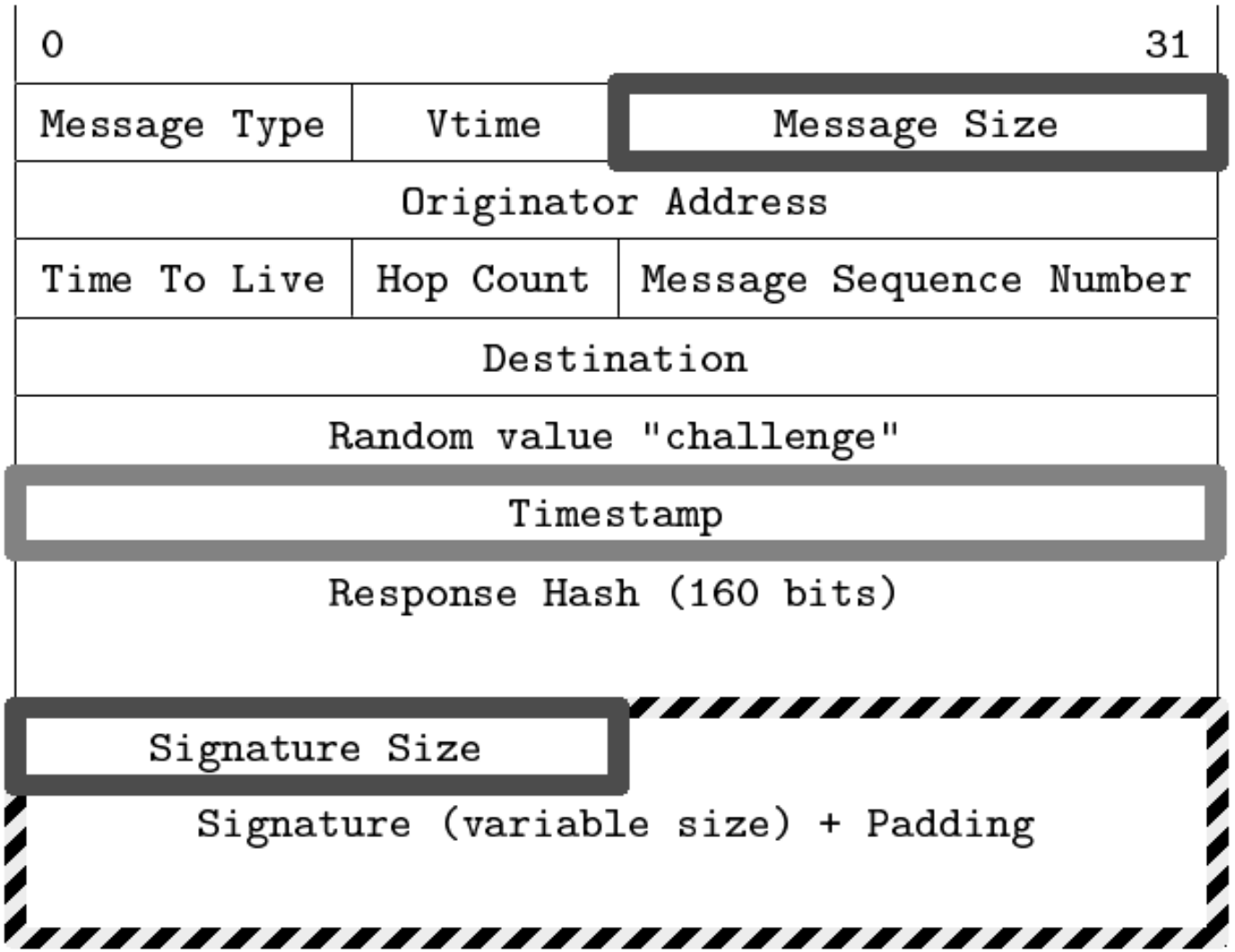
0		31
Message Type	Vtime	Message Size
Originator Address		
Time To Live	Hop Count	Message Sequence Number
Destination		
Timestamp		
Response Signature (160 bits)		
Signature (160 bits)		

OLSR web of trust plug-in



- Basically, keep the same structure of the secure plug-in, but:
  - Use PGP (GnuPG) signatures (that have a variable size)
  - Use trust to add entries to different routing tables

OLSR web of trust plug-in



Trust-based routing

- When an entry has to be inserted in the routing table, this entry is added to the appropriate routing table, on a trust value basis
- Then add tc filter policy routing rules based on the user traffic's IP ToS field

Load on the CPU

- Asymmetric cryptography is too heavy (more than 1000 times slower)

	Symmetric	RSA	DSA	measure unit
Length of the key	128	1024	1024	bits
Basic signature verification	0.132	159.422	150.784	milliseconds
Basic signature adding	0.05	55.477	60.130	milliseconds

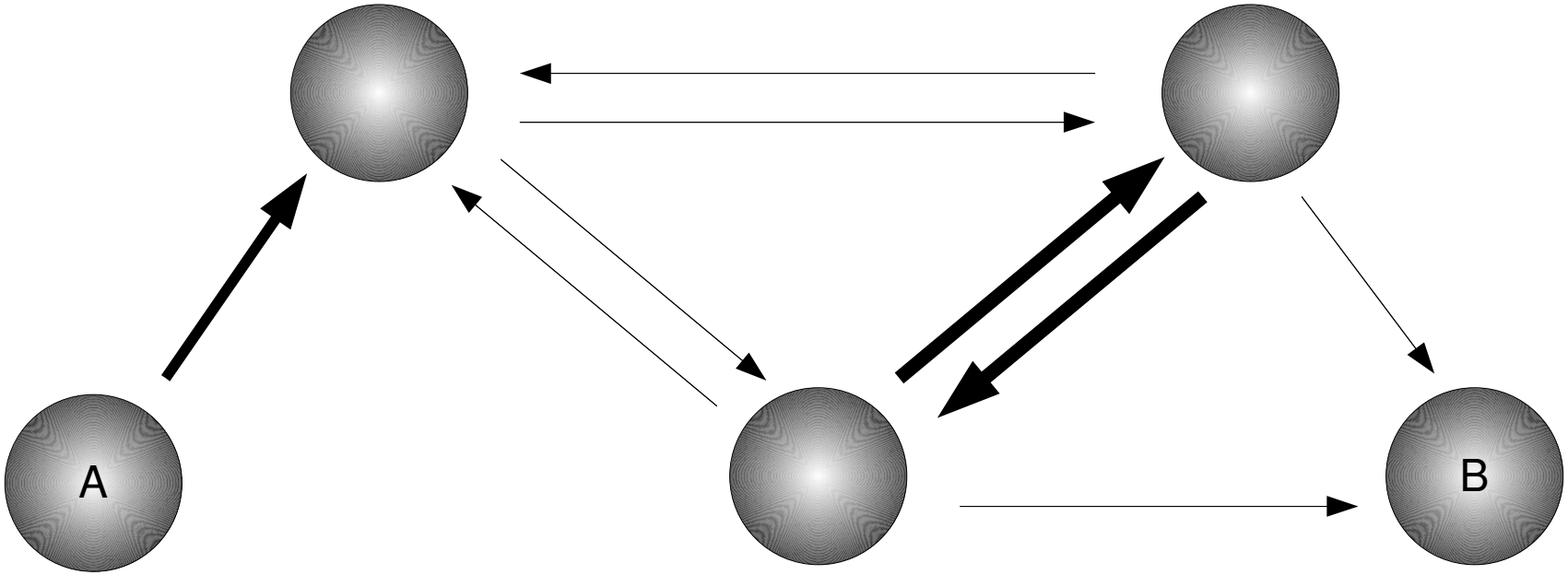
  - To use the plug-in, {Hello,TC,MID,HNA}Interval and {Hello,TC,MID,HNA}ValidityTime have to be increased

Load on the CPU

- PGP keys can be used to exchange symmetric session keys between neighboring nodes
  - This can be done during the timestamp exchange

Routing loops

- PGP trust is **not** a good metric (if decisions are made hop by hop)



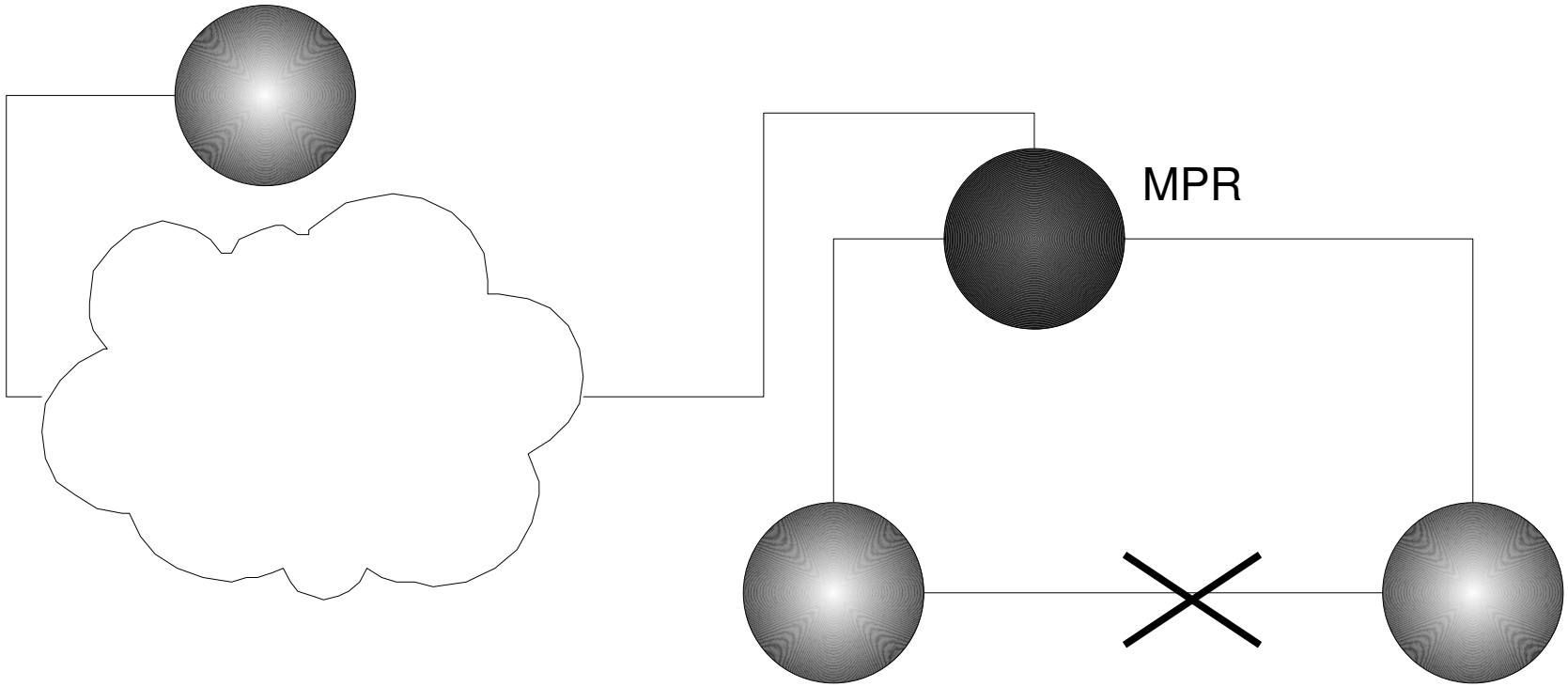
  - Solution: use a symmetric metric, e.g. Eigentrust

False Multipath routing

- When an entry has to be inserted in the routing table, this entry is added to the appropriate routing table, on a trust value basis, checking the “trust status”
  - This is like “tagging” routes
  - Which is the orthodox way of doing this?
    - Keep different OLSR topology databases?
    - Modify dijkstra with weights?

Some routes are missing

- It is not true that all nodes know all the topology



References

- http://hg.ninux.org/olsrd-ninux-messy/raw-attachment/wiki/WikiStart/Tesi_di_Claudio_Pisa_rc6_Trusted_Routing_In_OLSR_MANETs.pdf
- http://hg.ninux.org/olsrd-ninux-messy/raw-attachment/wiki/WikiStart/olsrd_weboftrust_plugin-0.1.tar.gz