IAIK TU Graz

SECRICOM
Seamless Communication for Crisis Management

Secure Multi-Agent System for Multi-Hop Environments

Stefan.Kraxberger@iaik.tugraz.at
Peter.Danner@iaik.tugraz.at
Daniel.Hein@iaik.tugraz.at

SCOS

SECRICOM

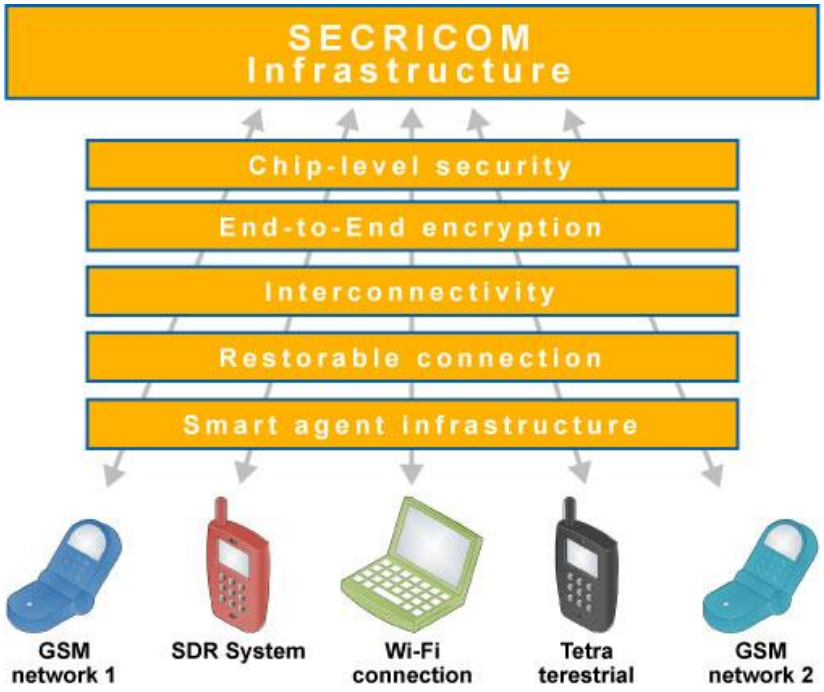
- SECRICOM is a research project creating Seamless Communication for Crisis Management for EU safety.

... a system that ensures end-to-end secure transmission of data and services across heterogenous infrastructures with real time detection and recovery capabilities against intrusions, malfunctions and failures ...

SECRICOM Vision

- Ability for responders to operate across different European emergency services/responder agencies as one cohesive unit at the time of crisis-level emergency

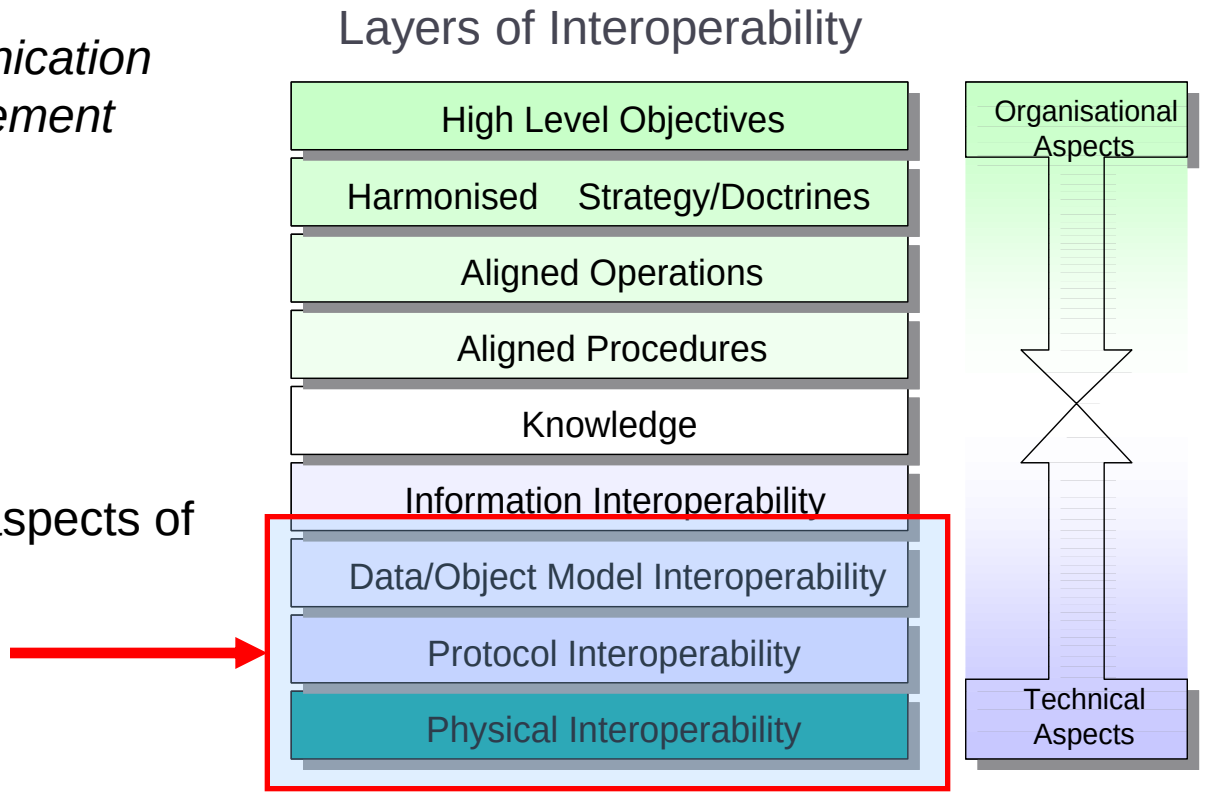
- Secure infrastructure for communication during a crisis with technical interoperability built into the design



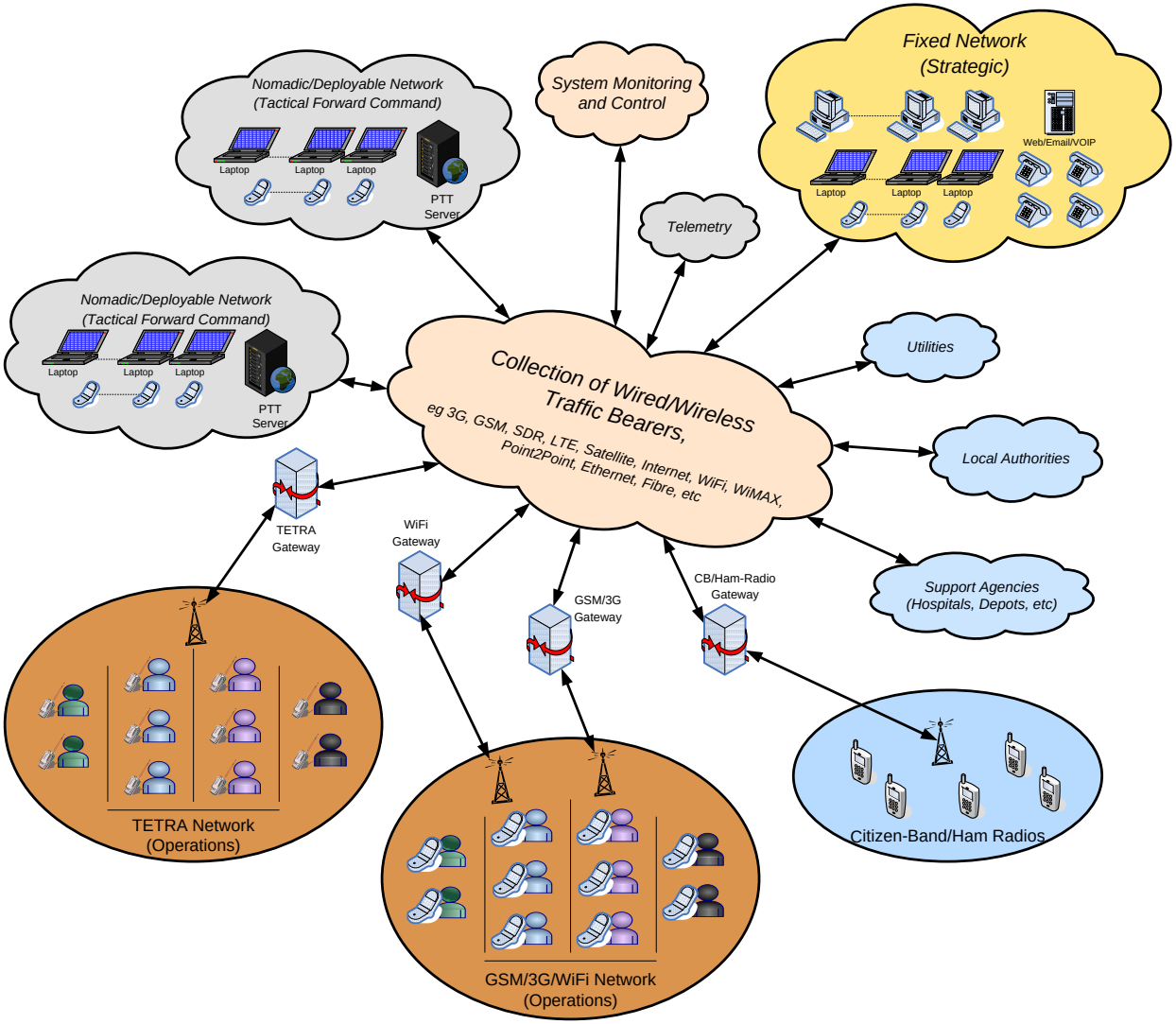
SECRICOM

Seamless Communication for Crisis Management

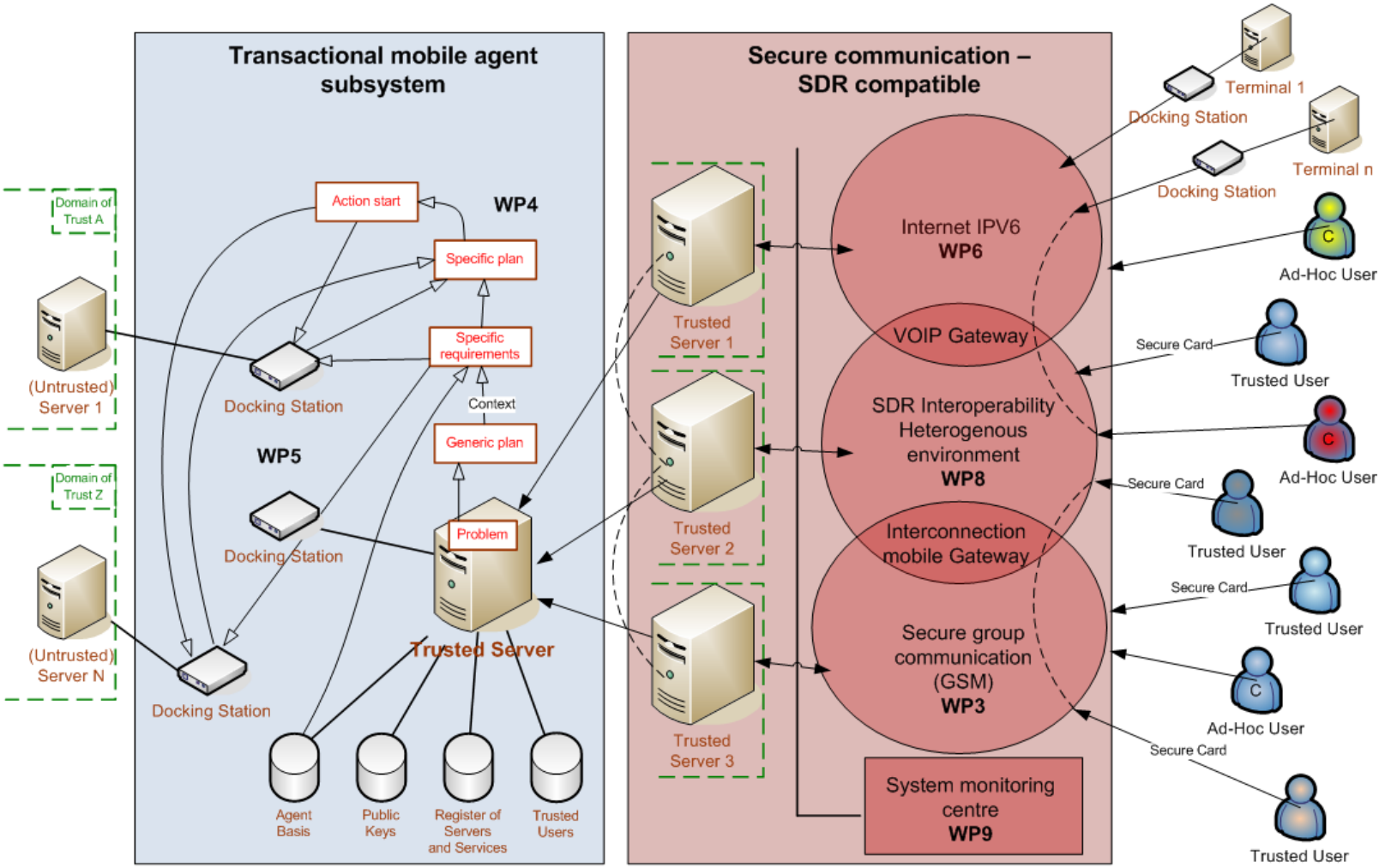
Scope: The technical aspects of Interoperability



SECRICOM High Level View



SECRICOM Technical View



Security Requirements

- The life cycle of a SECRICOM agent sets the following requirements on the security infrastructure of SECRICOM:
- It must have its own private key; this key must not be known to any other entity during the whole lifetime of the agent.
- It must be audited before it can be used; the audit must ensure, that the agent does only what its creator states it should do, and that it does not contain any malicious code
- It must be protected at all times from revealing its private key; it must always be either stored in a trusted device, or encrypted when it is outside of such device.
- Each audited agent must be issued a certificate, signed by its auditor, which states the capabilities of the agent as specified by its creator and verified by the auditor.

Security Requirements

- The infrastructure must contain a service for storing and accessing certificates of entities inside the infrastructure
- All results produced by a software agent must be protected from being revealed to any entity different than their intended recipient – the client. Also it must be asserted that their authenticity can be verified by the client upon their reception.
- The results provided by an agent must be signed by both the agent and the device running the agent's code in order to ensure the trust in the results by the Process Management System. Each secured device must be connected to SDM providing the encryption keys authenticating the device and its user, respectively.

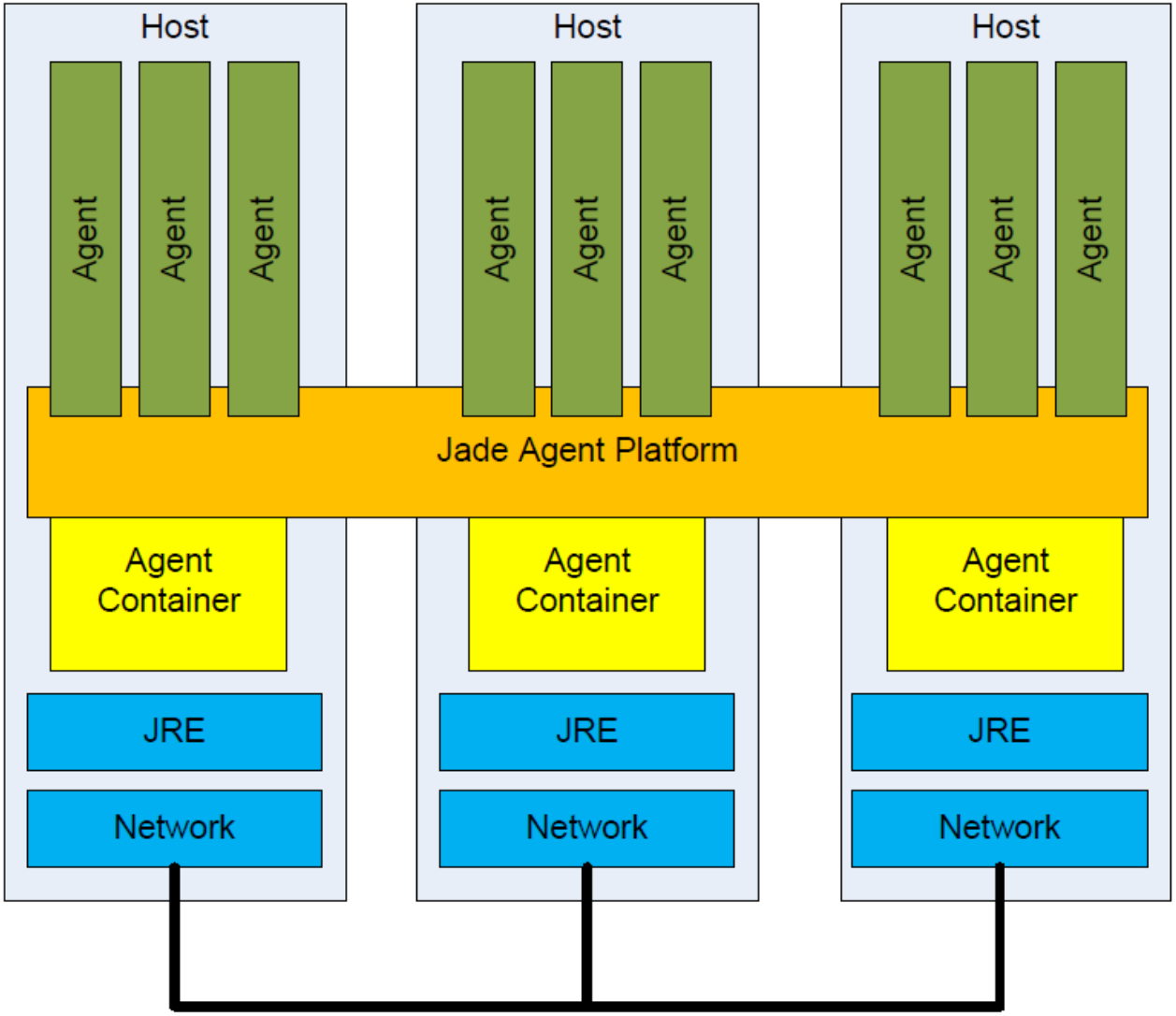
JADE - Communication Model

- Agents send/receive Java objects, that represent ACLMessages, within the scope of interaction protocols
  - JADE hides all message coding (encoding/parsing)
    - Envelope level
      - String-based, XML-based
    - Agent Communication Language level
      - String-based, XML-based, bit-efficient
    - Content Language level
      - FIPA SL-0 + API to register user-defined content languages
      - support for Base64-encoded direct Java object serialization
    - Ontology level
      - FIPA-Agent Management; JADE Agent Management
      - API to register user-defined content languages
    - the framework can be extended by users
      - all levels provide APIs to implement/register new codecs
      - work is in progress to improve CL and ontology level extendibility
  - JADE provides a library of common interaction protocols
    - users just need to implement the handle methods
    - users can compose agent tasks like super-states of FSM

JADE - Agent Mobility

- JADE supports intra-platform mobility and cloning
  - A platform can be distributed across multiple hosts
    - each host is an agent container
  - Agents can migrate between containers
  - Agents can clone across containers
  - Self-initiated
    - doMove(Location) / doClone(Location, String)
    - before/afterMove() before/afterClone()
  - Requested to the platform (via the AMS)
    - Fipa-request interaction protocol
    - jade.domain.MobilityOntology defines all the concepts and actions needed to support agent mobility and cloning

Java Agent DEvelopment framework



Secure Communication

- JADE doesn't provide security, but allows to make use of Java security features
- Each agent would need to implement these security features on their own
- Not possible to address all the SECRICOM security requirements

- SOLUTION: Use a common security overlay beneath the JADE layer
- We are using an secure P2P overlay called SePP which is now available from sourceforge

Secure Communication

- SePP provides a scalable secure P2P overlay for heterogeneous multi-hop environments
- Includes features such as:
  - Secure P2P management (join, leave, search, create and join groups,…)
  - Secure P2P overlay establishment and maintenance (route discovery, route update, route error)
  - Secure P2P communication with different security levels depending on the capabilities and requirements of the nodes
- Is implemented in Java and can be using an simple API

SePP and Key Management

- Different security levels are specified and can be used by the nodes
- The different security levels require different cryptographic keys and authentication information
- This keys are then used in the secure protocols of SePP
- Keys are managed by a hardware device called SDM

$$
\begin{aligned}
&S : h_0 = MAC_{SK}(Request, S, D, id, t_a)\\
&S \rightarrow *\ \ (Request, S, D, id, t_a, h_0)\\
&A : h_1 = MAC_{SK}(Request, S, D, id, t_a, A)\\
&A \rightarrow *\ \ (Request, S, D, id, t_a, A, h_1)\\
&B : h_2 = MAC_{SK}(Request, S, D, id, t_a, A, B)\\
&B \rightarrow *\ \ (Request, S, D, id, t_a, A, B, h_2)\\
&C : h_3 = MAC_{SK}(Request, S, D, id, t_a, A, B, C)\\
&C \rightarrow *\ \ (Request, S, D, id, t_a, S_{PS}, A, B, C, h_3)\\
&D : h_r = MAC_{SK}(Reply, D, S, id, t_a, A, B, C)\\
&D \rightarrow C\ \ (Reply, D, S, id, t_a, A, B, C, h_r)\\
&\dots\\
&A \rightarrow S\ \ (Reply, D, S, id, t_a, A, B, C, h_r)
\end{aligned}
$$

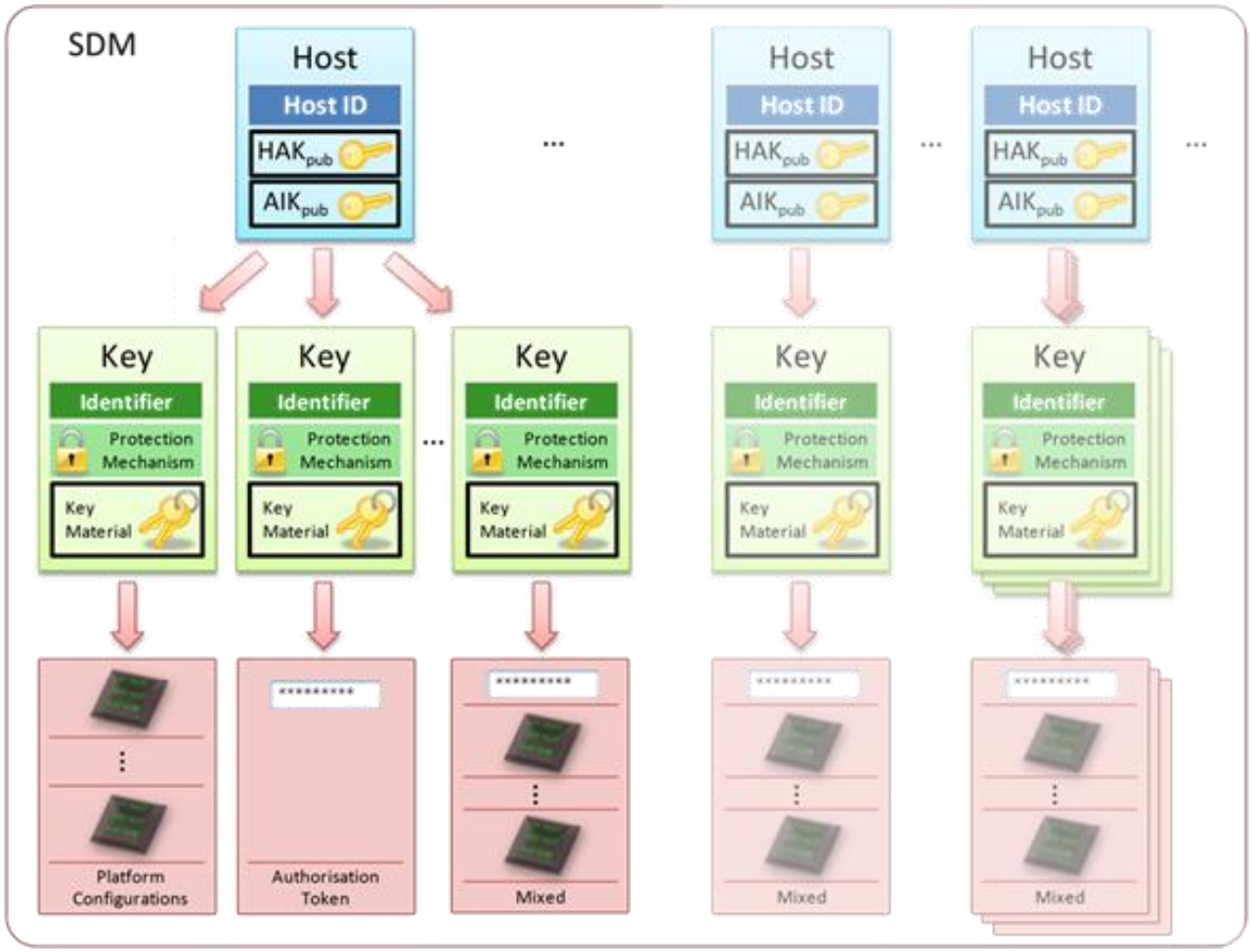
Implementation

		Security aspect		
		Admission security	**Data security**	**Session key protection**
Security level	**2**	Public/private key pair	Message auth. & encryption	Key refreshing & SCA countermeasures
	1	Pre-shared secret key	Message authentication	Key refreshing
	0	None	None	None

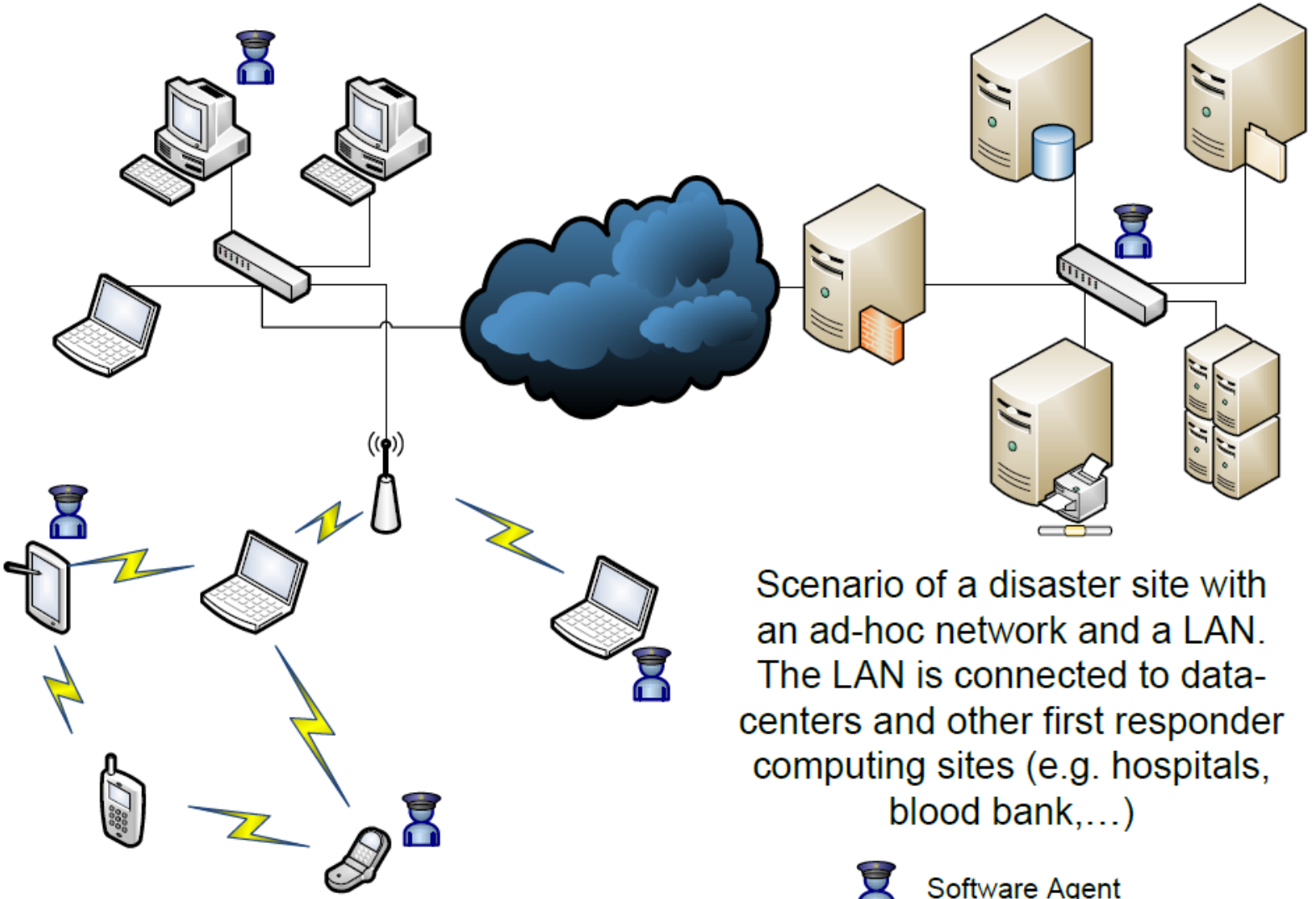
Secure Docking Module

- The Secure Docking Module (SDM) must provide two interlocked capabilities.
- On the one hand it is a secure key storage, while on the other hand it is a local attestation device.
- By establishing the trusted state of the host device it is possible to leverage the computing power of the host. Therefore the SDM does not provide an on chip computing environment.

Secure Docking Module



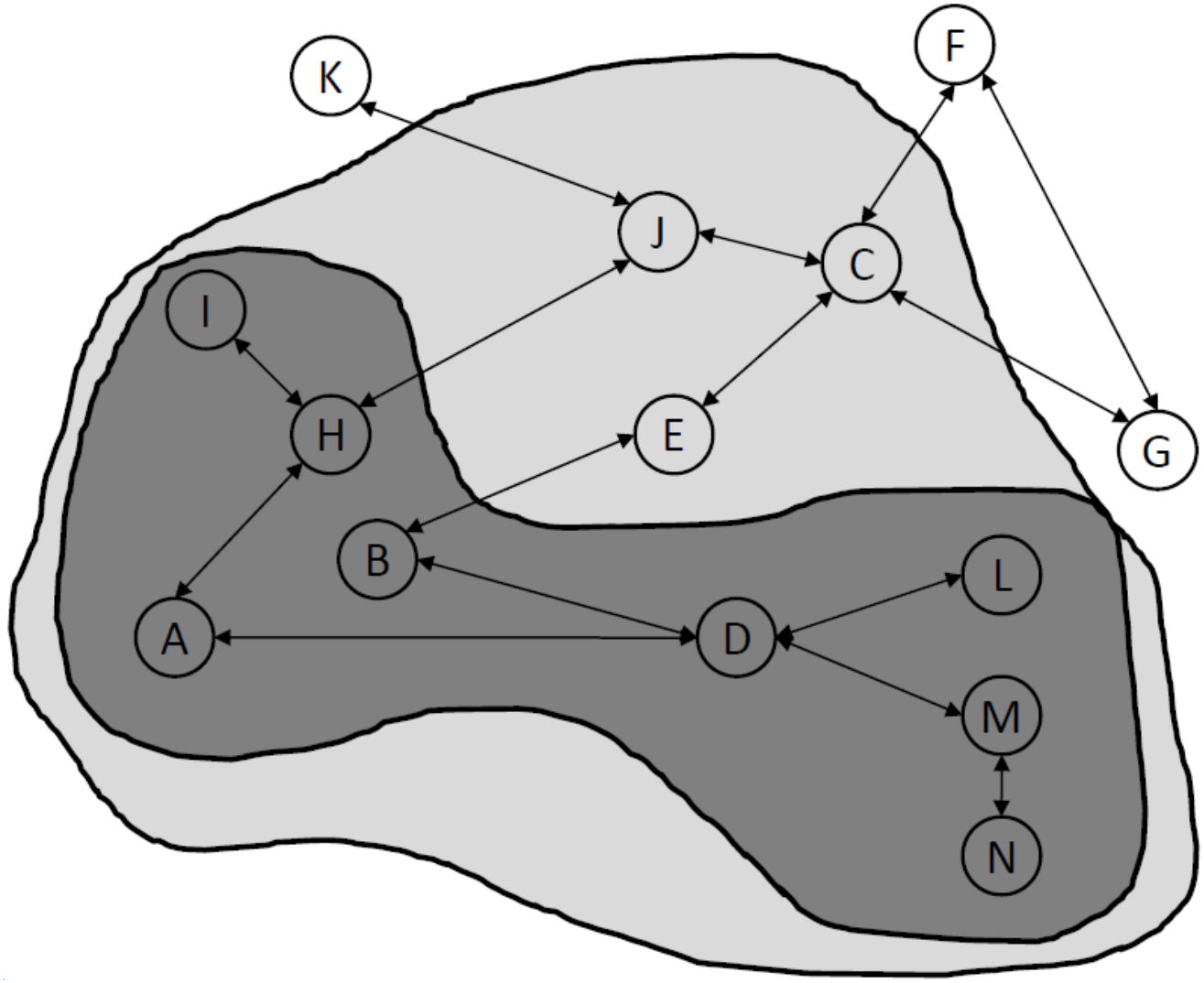
Secure Multi-Agent Systems



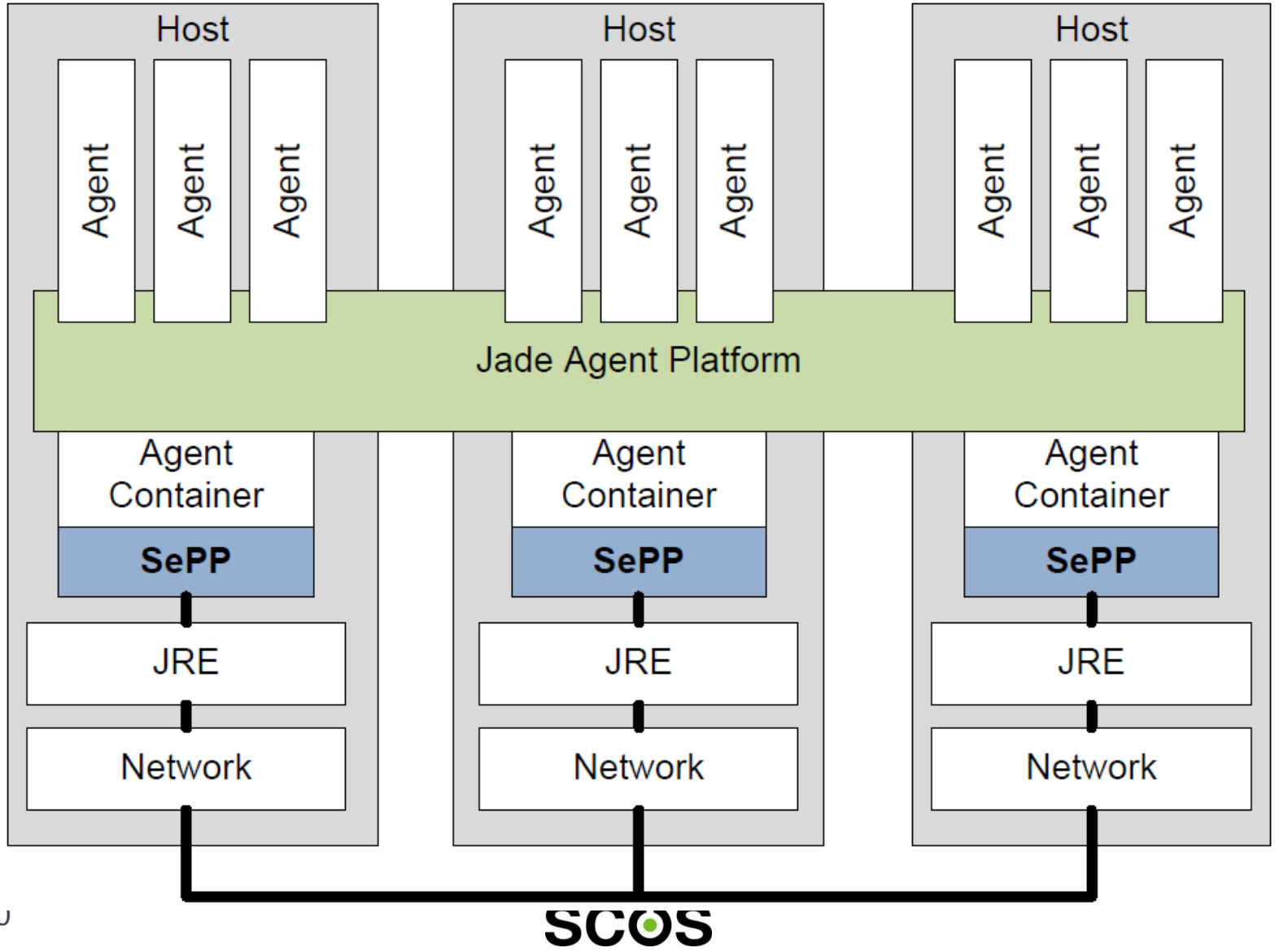
Scenario of a disaster site with an ad-hoc network and a LAN. The LAN is connected to data-centers and other first responder computing sites (e.g. hospitals, blood bank,…)

Software Agent

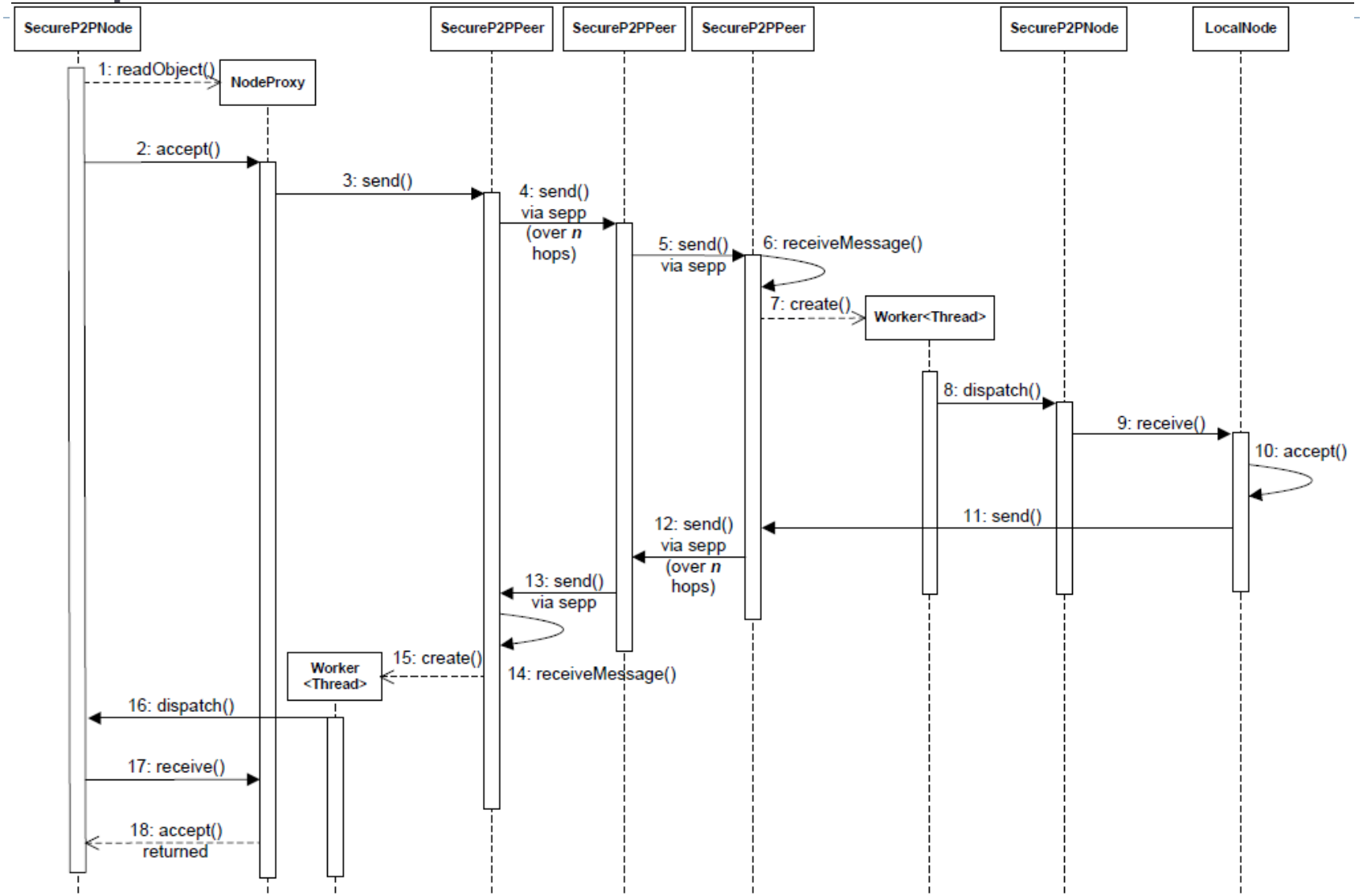
Security Level View of MAS



Implementation



Implementation



Evaluation

- The security levels and SePP are designed in such a manner that in level low everybody can participate. No guarantees on the security can be given.
- In security level medium shared secret keys are used for node authentication. It provides protection from outsider attacks.
- In security level high digital signatures are used to secure the established routes. Insider attacks from non-collaborating peers can be prevented.
- If peers collaborate they can always perform a wormhole attack by tunneling the requests from one peer to the other.

Evaluation

- To benchmark our implementation we compared it with JADE's out-of-the-box RMI implementation.
- Time measured from the start of the "Party" example until all messages have been sent, and the party has officially ended.
- ~7000 messages

	RMI [s]	SePP (direct) [s]	SePP (1 hop) [s]
1	4.40	15.40	16.20
2	3.70	14.80	16.90
3	3.40	14.30	15.90
4	3.60	16.10	15.80
5	3.50	14.00	15.40
6	3.90	14.60	16.10
7	3.40	13.40	15.40
8	3.70	15.00	15.80
9	3.30	16.00	15.30
10	3.40	14.30	16.00
Mean	3.63	14.75	15.88

Conclusion

- The proposed multi-agent system with SePP as underlying communication infrastructure enables the use of agent technology in multi-hop environments
- It provides simple means of integrating and enhancing existing MAS with secure communication mechanisms without the need for redesign or re-implementation of the MAS itself.
- We introduced an RMI-style interaction layer which mediates between the MAS on top of a secure P2P framework.
- The security management is separated from the MAS application and can be adjusted according to the needs of the participating entities.
- With our solution it is possible to comply with various security requirements in a fine grained manner since it is possible to select security levels from a global to a group scale.

Thank you for your attention