Security in TripCom

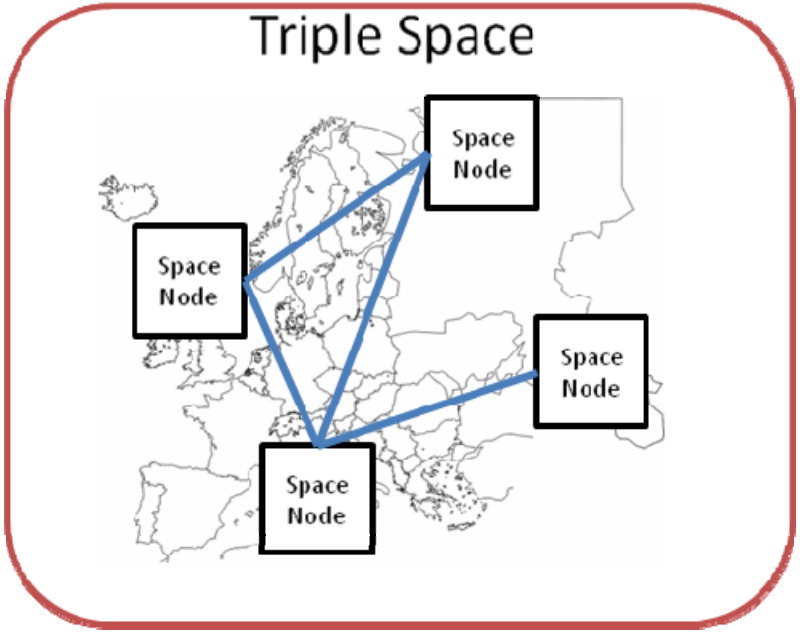
TripCom meeting
Berlin, 18-19 January 2007

Alessandro Ghioni & Davide Cerri
CEFRIEL – Politecnico di Milano
{ghioni, cerri}@cefriel.it

universität innsbruck
National University of Ireland, Galway
Universität Stuttgart Germany
TU WIEN Technische Universität Wien Vienna University of Technology
Freie Universität Berlin
ontotext Semantic Technology Lab
profium
CEFRIEL Forging Innovation
Telefónica Telefónica Investigación y Desarrollo

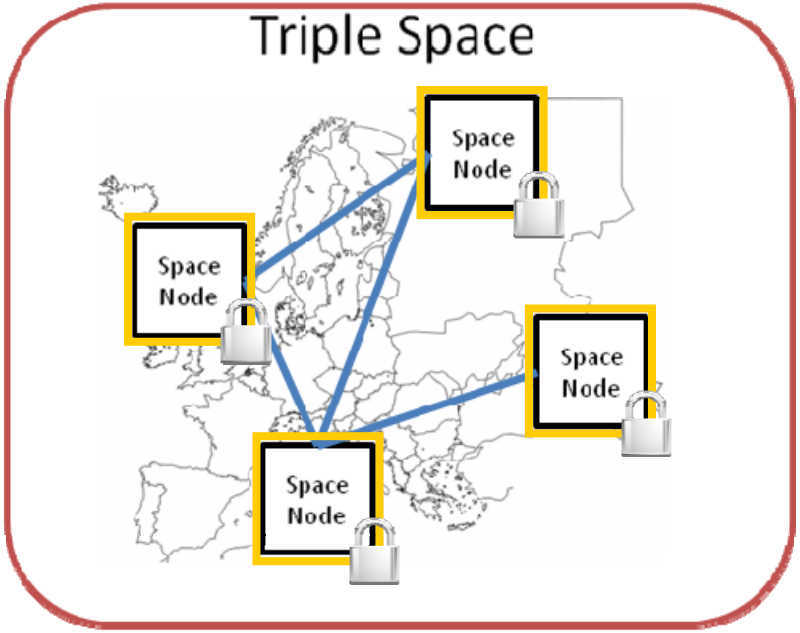
Triple space nodes

- The Triple Space is a distributed infrastructure consisting of a **set of nodes**.
  - There is **no central authority** or point of control for the “global” Triple Space.
  - Each node is managed by **its own authority**.
  - There may be **agreements** between different nodes / authorities (federation etc.)



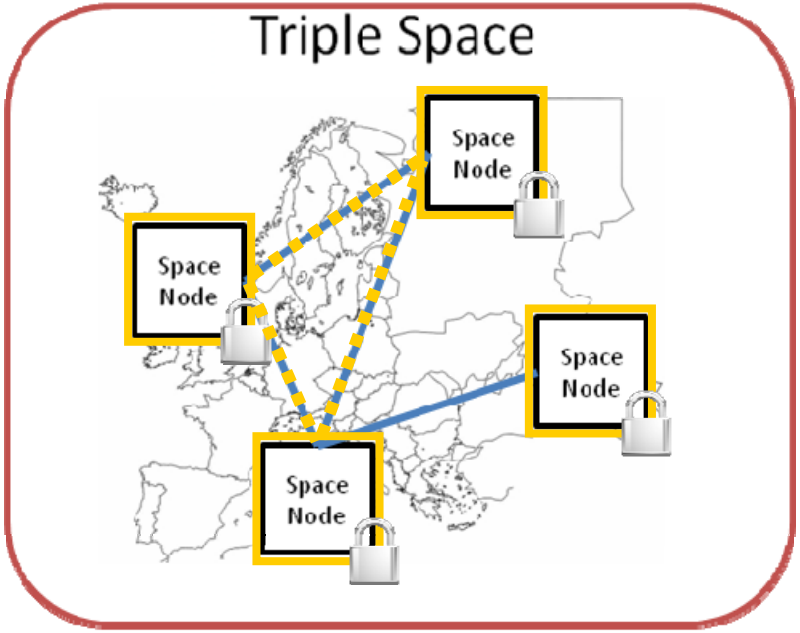
Security perimeters: Triple Space

- Since each node may be operated by an independent authority, node boundaries are **security perimeters**.
  - The system must be seen as a set of a space nodes, and not just as a global space.
  - **Security must be enforced when information crosses nodes boundaries**, because there is a transfer towards different authorities.



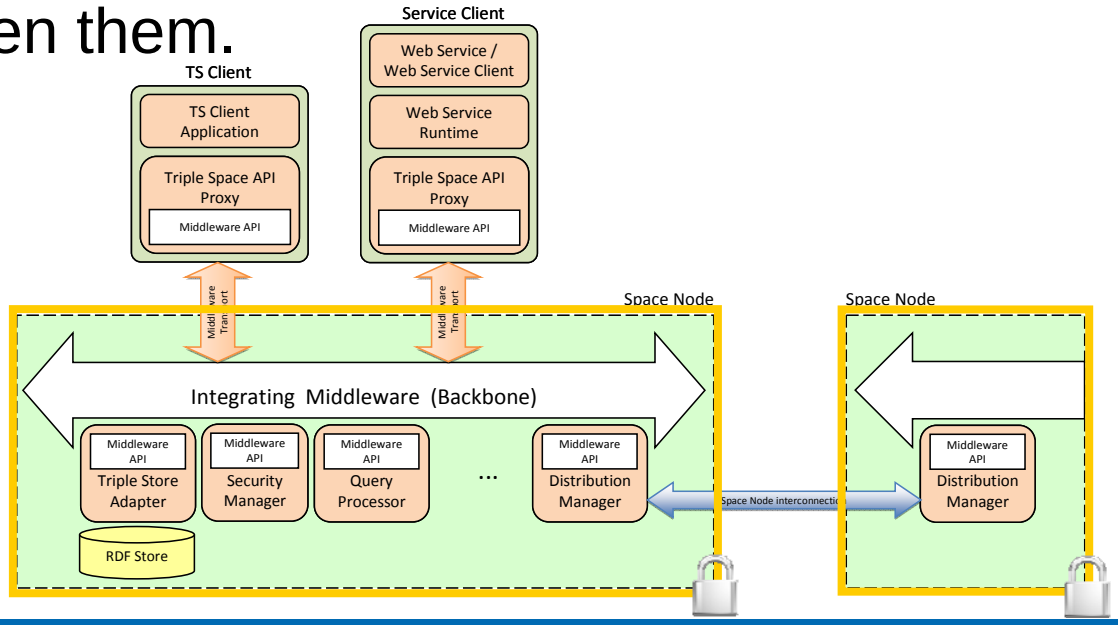
Security perimeters: trust relationships

- Agreements between different authorities result in **trust relationships** between different nodes.
  - Nodes with trust relationships are not totally unknown, and therefore can be granted some rights.
  - A trust relationship is not the same as a security perimeter, but can be seen to “extend” it to some extent.
  - We cannot assume that all nodes will have some trust relationship with all other nodes.



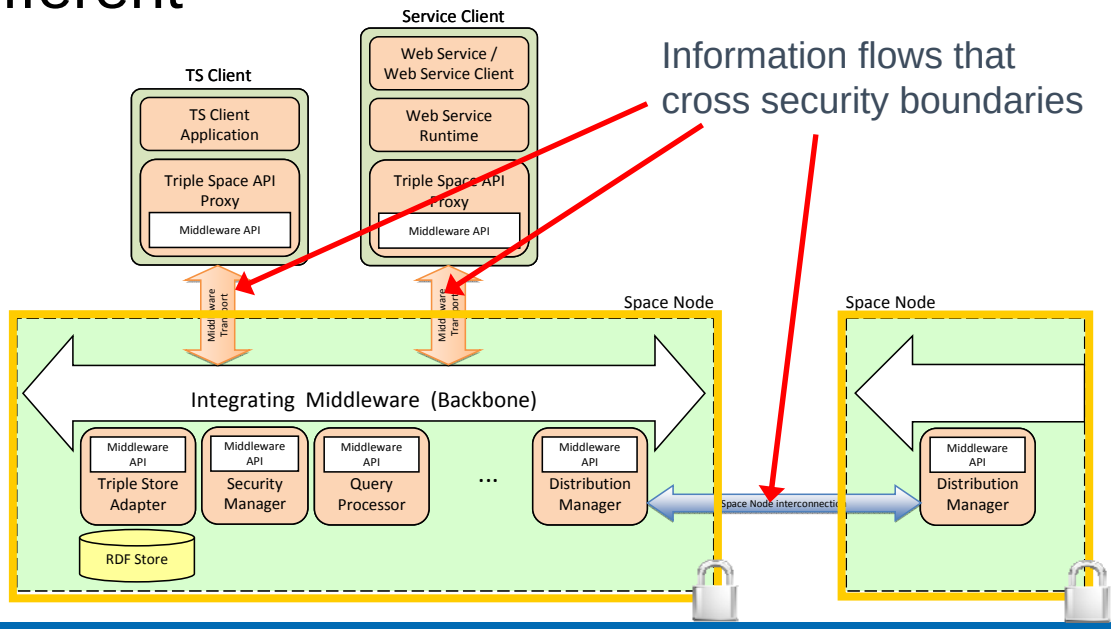
Security perimeters: single node

- Node components are **within** the security perimeter.
  - No trust issues between components of a single node: components are trusted by definition.
  - If components are **physically distributed**, the node authority/administrator must ensure communication security between them.



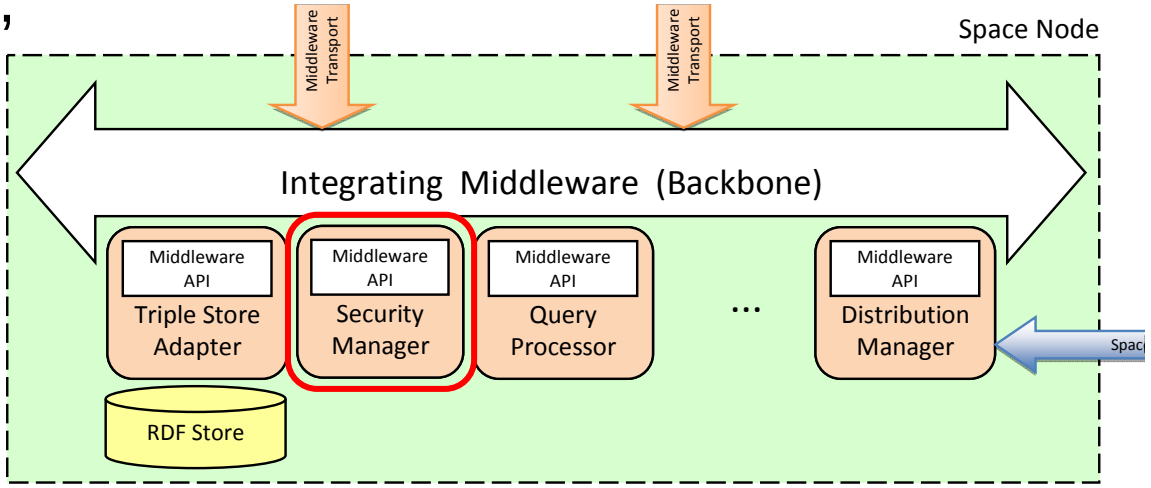
Security perimeters: single node

- The security perimeter is crossed when **a client talks with a node**.
- The security perimeter is crossed when **a node talks with another node**.
  - Data cannot be moved “transparently” between nodes under different authorities.

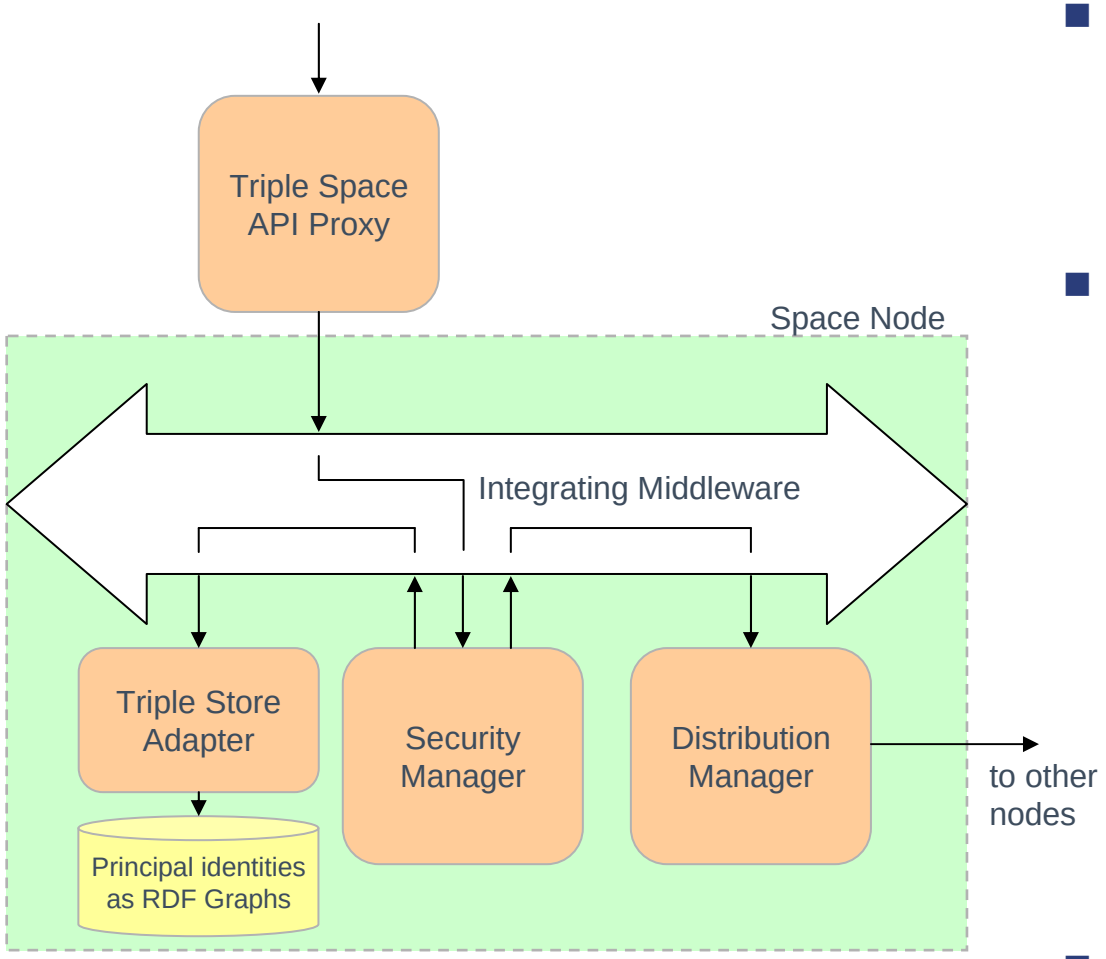


The security manager component

- The **security manager** is one of the node architectural components.
- It implements the **policy decision point**, while the policy enforcement point is implemented by the integrating middleware.
- Its main functionalities are:
  - **authentication**,
  - **authorization**.



Authentication



- An **authentication request** is received via the TS API component and is written into the space.
- The security manager picks up the request and:
  - either provides response using **information stored in the local triple store** (if the agent is locally known),
  - or asks the distribution manager to **forward the request to other nodes** (if the agent is not locally known).
- The security manager acts like an **identity provider**.

Authentication

- If the agent requesting access is not locally known, **other entities** must be involved:
  - external authentication services (e.g. CAs),
  - other nodes.
- If the request must be forwarded to other nodes, we need to know **which is the right node** and how to reach it.
  - This is a subset of the general TS distribution problem.
  - We could use some peer-to-peer technology.
- Some **standards** that can be reused:
  - SAML (Security Assertion Markup Language),
  - so-called “identity 2.0” solutions (e.g. OpenID).

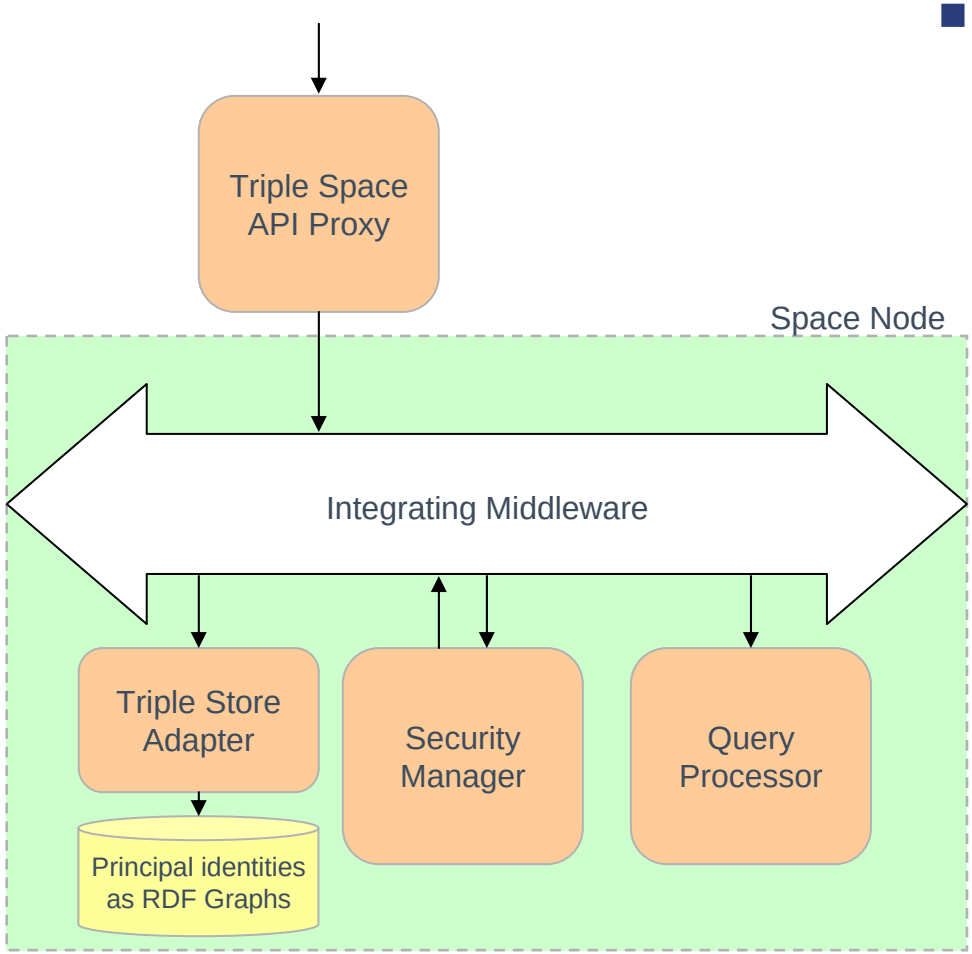
Authentication

- If the authentication request must be routed to other nodes, **other authorities** (node owners) are involved.
  - We cannot assume that all nodes, and therefore all authorities, are known a priori.
    - We cannot assume that all trust relationships are known a priori, and that previously unknown nodes cannot be trusted.
  - Nodes will need to decide whether to trust another node as identity provider maybe having only **partial knowledge**.
    - **Hierarchical** trust solutions may be used in closed worlds.
    - **Reputation** could be used in an open world.

Authorization

- Access control deals with **subjects**, **objects**, and **permissions**.
  - A subject is authorized to perform an action on an object.
- What are these in TripCom?
  - In the TS ontology we have objects such as triples, graphs, and (sub)spaces.
  - In the TS API we have actions such as read and create space, upon which we can define permissions.
- So we can have access control rules such as:
  - agent *X* can read graph *G1*
  - agent *Y* can create subspaces in space *S1*

Authorization



- The security manager is a node component like others and **communicates with others through the space** (maybe using some sort of privileged access).
  - A request is received via the TS API component, is translated into a tuple/graph and is written into the space.
  - The security manager takes care of any incoming request, and validates its access rights.
  - Other components (e.g. the query processor) consider only requests that have been validated by the security manager and carry a proper authorization “mark”.

Authorization

- Related **standards** and approaches:
  - RBAC (Role-Based Access Control),
  - XACML (eXtensible Access Control Markup Language).
- We would like to **reuse existing standards** leaving the door open for **future "semantic security" extensions** (out of scope in TripCom).
  - Storing access control information using triples.
  - Defining an ontology for this information.

Sixth Framework Programme Priority 2
Information Society Technologies (IST)
Specific Targeted Research Project

Authorization

- Applications will deal with higher level objects and functions, and would like to define **higher level policies** based on these.
- Just like the translation of higher level functions into TS operations must be performed by the application, in the same way the application must **translate** higher level policies into TS policies, which are based on TS concepts.
  - The TS can support the application with some abstractions, e.g. **transactions** (a transaction must be authorized as a whole) and **roles** (agents can play roles and be granted related permissions).

Workplan

- We now focus on **building the security manager** component.
  - T5.2 (ends M18): design phase;
    - definition of data models, coordination with other components, possible extensions to TS ontology and API, reuse of standards, etc.
  - T5.3 (ends M24): first implementation.
- Security model for the **multi-node environment** will follow (depends on decisions about distribution in the whole infrastructure).