TripCom

Triple Space Communication

FP6 – 027324

Deliverable

D5.1

Security and trust requirement analysis and state-of-the-art

Dario Cerizza
Davide Cerri
Alessandro Ghioni
Geri Joskowicz
Jacek Kopecky
Daniel Martin
Henar Muñoz
Lyndon Nixon
Noelia Pérez
Thorsten Scheibler
Daniel Wutke

October 5, 2006

Executive Summary

This deliverable aims at discussing what "security" means in TripCom, assessing requirements and state-of-the-art. After a discussion about the presence and role of "authorities" ruling the triple space, and about the impact of this on security requirements, the document analyses state of the art for security of TripCom infrastructure and for trust and reputation, and discusses security issues and requirements from EAI and e-health use cases.

Document Information

IST Project Number	FP6 – 027324	**Acronym**	TripCom
Full Title	Triple Space Communication		
Project URL	http://www.tripcom.org/		
Document URL			
EU Project Officer	Werner Janusch		

Deliverable	**Number**	5.1	**Title**	Security and trust requirement analysis and state-of-the-art
Work Package	**Number**	5	**Title**	Security and Trust

Date of Delivery	**Contractual**	M6	**Actual**	5 Oct 2006
Status	version 2		final ☒	
Nature	prototype ☐ report ☒ dissemination ☐			
Dissemination Level	public ☒ consortium ☐			

Authors (Partner)	Dario Cerizza (CEFRIEL), Davide Cerri (CEFRIEL), Alessandro Ghioni (CEFRIEL), Geri Joskowicz (TUW), Jacek Kopecky (LFUI), Daniel Martin (USTUTT), Henar Muñoz (TID), Lyndon Nixon (FUB), Noelia Pérez (TID), Thorsten Scheibler (USTUTT), Daniel Wutke (USTUTT)			
Resp. Author	Alessandro Ghioni		**E-mail**	ghioni@cefriel.it
	Partner	CEFRIEL	**Phone**	+39 02 23954 212

Abstract (for dissemination)	This deliverable aims at discussing what "security" means in Triple Space Computing, assessing trust and security requirements and state-of-the-art.
Keywords	triple space computing, security, trust, reputation, tuple space security, middleware security, authentication, authorization, authorities, enterprise application integration, e-health

Version Log			
Issue Date	**Rev No.**	**Author**	**Change**
15 Sep 2006	1	All	First complete version
5 Oct 2006	2	All	Final version

Project Consortium Information

Acronym	Partner	Contact
Leopold Franzens University Innsbruck http://www.deri.at	LFUI	Prof. Dr. Dieter Fensel Digital Enterprise Research Institute (DERI) Innsbruck, Austria E-mail: dieter.fensel@deri.org
National University of Ireland, Galway http://www.deri.ie	NUIG	Dr. Laurentiu Vasiliu Digital Enterprise Research Institute (DERI) Galway, Ireland Email: laurentiu.vasiliu@deri.org
University of Stuttgart http://www.iaas.uni-stuttgart.de/	USTUTT	Prof.Dr. Frank Leymann Inst. für Architektur von Anwendungssystemen (IAAS) Stuttgart, Germany E-mail: frank.leymann@informatik.uni-stuttgart.de
Vienna university of Technology http://www.complang.tuwien.ac.at/	TUW	Prof.Dr. eva Kühn Institut für Computersprachen Vienna, Austria E-mail: eva@complang.tuwien.ac.at
Free University Berlin http://www.ag-nbi.de/	FUB	Prof. Dr.-Ing. Robert Tolksdorf AG Netzbasierte Informationssysteme Berlin, Germany E-mail : tolk@inf.fu-berlin.de
Ontotext Lab, Sirma Group Corp. http://www.ontotext.com/	ONTO	Atanas Kiryakov, Vassil Momtchev, Ontotext Lab, Sirma Group Corp. Sofia, Bulgaria E-mail: vassil.momtchev@ontotext.com
Profium OY http://www.profium.com/	Profium	Dr. Janne Saarela Profium OY Espoo, Finnland E-mail: janne.saarela@profium.com
CEFRIEL SCRL. http://www.cefriel.it/	CEFRIEL	Davide Cerri CEFRIEL SCRL. Milano, Italy E-mail: cerri@cefriel.it
Telefonica I+D http://www.tid.es/	TID	Sara Carro Martinez Telefonica I+D Madrid, España E-mail: tripcom@tid.es

Table of Contents

List of Abbreviations

AAA	Authentication, Authorization, Accounting
AH	Authentication Header
ACL	Access Control List
DAC	Discretionary Access Control
DBMS	Database Management System
ESP	Encapsulating Security Payload
GSSAPI	Generic Security Services Application Program Interface
HTTP	Hyper Text Transfer Protocol
HTTPS	HTTP over TLS/SSL
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IMAP	Internet Message Access Protocol
IP	Internet Protocol
IPsec	Internet Protocol Security
LKH	Logical Key Hierarchy
MAC	Mandatory Access Control
MAC	Message Authentication Code
OASIS	Organization for the Advancement of Structured Information Standards
PGP	Pretty Good Privacy
POP	Post Office Protocol
PKI	Public Key Infrastructure
RBAC	Role Based Access Control
RFC	Request For Comment
RPC	Remote Procedure Call
SAML	Security Assertion Markup Language
S/MIME	Secure/Multipurpose Internet Mail Extensions
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SPNEGO	Simple and Protected GSSAPI Negotiation Mechanism
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TESLA	Timed Efficient Stream Loss-tolerant Authentication
TLS	Transport Layer Security
TS	Triple Space
VPN	Virtual Private Network
W3C	World Wide Web Consortium
WSMO	Web Service Modeling Ontology
WTS	World of Triple Spaces
WWW	World Wide Web
XML	eXtensible Markup Language

1 Introduction

In a potentially open and decentralized infrastructure, used by many different and often unrelated actors, security issues are a crucial aspect. Therefore, providing suitable security mechanisms may be a very important success factor for TripCom.

Objectives of "security and trust" workpackage include assuring security and privacy of the information written in the triple space, providing the theoretic foundations of a mechanism for establishing end-to-end trust between dynamically discovered actors, and provide a prototypical, pluggable implementation of these.

This document is the first deliverable of security and trust workpackage, and aims at discussing what "security" means in TripCom, assessing requirements and state-of-the-art. This is needed in order to provide input for the subsequent design tasks of TripCom security model and mechanisms, but it is also useful in order to clarify security key concepts and needs that the whole project must be aware of, because security cannot be considered an "add-on" feature.

The document is divided into three main parts. The first part (chapter 2) deals with the concept of "authority" and how this impacts on TripCom. It introduces the concept of the "world of triple spaces", differentiating it from the "single" triple space, and discusses different aspects of security requirements in the two cases. The second part (chapters 3 and 4) deals with state-of-the-art for, respectively, security of the infrastructure (storage, tuple space, middleware) and trust and reputation issues. The third part (chapters 5 and 6) discusses security issues and requirements from TripCom use cases: Enterprise Application Integration and e-Health. Finally, a brief summing-up of all security requirements is also provided (chapter 7).

2 Authorities and requirements

Dealing with security, we must consider the concept of *authority*, i.e., an entity in the system that establishes rules and policies that must be followed. Therefore, here we consider a *single triple space* (TS) as a space ruled by a single administrative "root" authority[1]. This authority may also delegate some administrative tasks to other entities, thus having a hierarchy of authorities, but there is still a *root*. Anyway, just like the World Wide Web can be considered a single entity but does not have such an authority, we must also deal with a *"world of triple spaces"* (WTS), which is the set of all the single triple spaces, each one ruled by its own authority. These authorities are administrative authorities, i.e., they set policies on their triple spaces, but they usually do not decide whether what is being published on the space is true or false, trustworthy or not (this issue is common to a single triple space and to the world of triple spaces).

Therefore, we can divide the security issues that arise in TripCom into three areas:

- security issues for a single triple space,
- security issues for the world of triple spaces,
- trustworthiness of information in the triple space.

2.1 Security issues for a single triple space

A single triple space has an *"owner"* who administers it (the TS root authority), and some *users* (services, applications, etc.) who can read and write information in the space. Drawing an analogy, we can see the TS owner as the administrator of a web message board (forum site): he has much more administrative rights than a normal user (e.g., he can create new forums, set forum topics, ban bad users, etc.), but he is not the author/owner of all information published in the space he controls. On the other hand, users may be able to read/write information (like posts in a forum) or even be delegated to perform some administrative tasks (like a forum moderator, thus there can be a hierarchy of authorities), but they must follow global policies set by the administrator.

Data published in the TS needs not to be public (i.e., only some users may be allowed to access certain data). This confidentiality requirement leads to classical AAA (authentication, authorization, accounting) requirements.

2.1.1 Authentication

In order to access a TS, users must be able to authenticate themselves, i.e. to give a proof of their identity. This is a prerequisite for other security requirements, and can probably be addressed with one of the many classical authentication solutions (passwords, PKIs, etc).

[1]This is a logical view, unrelated to a physical distribution of the triple space between different machines, or to different triple spaces being on the same machine.

2.1.2 Identity and anonymity

Depending on the TS policy, some operations could be performed by anyone: in this case, anonymous (i.e., unauthenticated) users may be admitted. An important question about authentication is what we mean with *identity*: the "virtual" identity (i.e., the identity used in the TS) may be something explicitly bound to a "real world" identity in some way, or it may be a pseudonym with no such link. Moreover, mechanisms could be required that provide delegation and impersonation services for the parties interacting through the triple space. Delegation and impersonation mechanisms often rely on concepts of role-based (RBAC) systems, since it makes more sense to impersonate to a role than to another user identity. Some applications may also need some degree of anonymity, i.e., the guarantee that the real identity of the user cannot be disclosed.

2.1.3 Authorization

Users are not equal, and different users will have different access rights. Security policies must state who can do what, and access control mechanisms must enforce these policies.

The first question that arises is what a user could be able to do on what. Therefore, we must identify the "atom" (i.e., the smallest data entity) on which access control policies can be defined, and the access rights (e.g., read, modify, delete, etc.) that can be granted on it. A single triple could be an atom, but requiring to define access rights on every single triple can be overwhelming. This atom could also be a sort of logical container, no matter how much data it contains and how it is internally organized, just like a file in a file system. Regarding the definition of single access rights, this clearly depends on which operations can be performed on triples and/or "atoms".

The second question is who decides who can do what. Each "atom" could have an owner who sets access rights on it; this would lead to a DAC model, in which information owners determine access control policy on their own data. The TS administrator can be seen as the owner of the space, so he can, for example, set policies about who can create new triples/atoms.

Experience with large data management systems teaches that fine-grained access control is impractical and leads quickly to enormous administrative overheads. The use of fine-grained access control requires that each process (or interaction, business task, etc.) accessing the data needs to be known and analysed a priori. This is nearly impossible even in well controlled intranet structures and merely unreasonable in Internet structures. This means that role-based authorization in conjunction with coarse grained access control could lead to useful solutions. In role-based authorization mechanisms, users are generally mapped statically to the set of roles they play, however in conjunction with reputation and trust mechanisms, users can be allowed to claim membership of a role, and that claim will be verified by evaluating its trustworthiness. See Section 2.3 for more trust considerations.

Security policies must be expressed using an appropriate language and format. The triple space should be secured using the triple space itself, therefore security policies should be expressed using triples. Actually the field of "security ontologies" is blooming: a DAML security ontology has been published that helps to describe security aspects of services,and additional ontologies have been proposed to describe

more generic resources. "Semantic security" is a very challenging topic, and expressing security properties in triples leaves the door open for extensions along that way.

Authorization is always at the intersection of the functional dimension ("allowed to sign a medical report") and the data dimension ("allowed to get the medical records of Mr. Brown"). Practical systems must find the right balance between these two dimensions. Consider the business rule: "In your own department you have the authority to sign an order up to 10.000 euro; in any other department you can sign up to 1.000 euro". The action ("to sign") on the resource ("an order") is protected by an authorization policy. The policy is expressed in rules that know about the principal, the function and the resource. The policy decision component must have some means to access the data ("order value"). The XACML standard from OASIS provides the language to express such policies. The standard proposes also a prototype architecture consisting of a PEP ("policy enforcement point"), and a PDP ("policy decision point") as the core elements. Fundamentally, the triple space provides some sort of publishing, retrieval / reasoning as well as an environment used by external services. The policies required to authorize the access to information in the triple spaces as well as the policies controlling the access to external services require information maintained in the triple space itself. It seems therefore natural to use the triple space to maintain policies and to use the capabilities of the triple space to reason about policies. The basic approach to apply authorization policies on semantic information systems, is to map XACML constructs into a semantic representation and to reason on these representations. This idea has been proposed by various authors and has been successfully demonstrated[2].

Regarding enforcement, the triple space infrastructure can enforce access rights on data (like a file system or a DBMS), but this usually means that "root" (the entity that administers the infrastructure) can bypass this mechanisms and do anything. Therefore, security can also be "embedded" in stored data, besides (or regardless of) access rights enforcement made by the infrastructure (e.g., users can encrypt data so that, even if it can be accessed by anyone, it can be "understood" only by those who have the right key). We envisage that both kinds of protection can be needed and should be supported. An important thing to note is that the second kind of protection may prevent the TS owner from doing some operations, since he cannot fully access some information published in the space he controls.

Different access rights mean that different users will have a different search set in the space, and thus will receive different responses to the same query. Users could be notified if a security policy prevented access to some information: for example, if a user submits a query and the results are biased because not all the triples can be accessed, the user may wish to be aware of this. However, such a feature may indirectly disclose information that should be protected, because it gives the user some hints about information he is not allowed to access. Therefore, in case this should be allowed carefully, and cannot be the norm.

2.1.4 Accounting and logging

The TS should be able to log what users do, in order to make them accountable of their deeds. Pseudonymity is usually compatible with this requirement, since pseudonyms are virtual identity information and they allow linking different actions to the same

[2]See [19] for an overview of the current implementations.

virtual identity, while complete anonymity is not. Note that here we intend accountability related to doing something "bad", like trying to break a policy, or attacking the TS infrastructure. We do not consider here accountability about publishing information that may be considered "wrong" by someone. If something "bad" is detected, the TS owner can take appropriate measures against the responsible user.

2.2 Security issues for the world of triple spaces

Some of the problems discussed above for a single triple space become much more difficult if we must address them in the world of triple spaces (WTS), which can be seen as analogous to the WWW in contrast to a single web site.

First of all, we can assume that each TS has an owner/administrator or a hierarchy of administrators (as discussed in the previous section), we can also assume that an explicit "federation" of multiple TSs has an owner (i.e., an entity that rules it), but we cannot assume that the WTS as a whole has an owner, just like the WWW hasn't one. This means that a global authority, which sets security policies and mechanisms for the WTS like a TS owner does for its own TS, cannot exist. The absence of a single authority that can establish rules and is globally trusted poses the problem of trust. Let's see how this problem impacts on the aforementioned requirements.

- *Authentication and identity*: basically, we must face two problems: multiple identities and trust. The first means knowing that the user `johndoe` on a TS and the user `jdoe` on another TS are the same entity. This can be possible if these identities are linked to a "real world" identity (which must be the same), or with an agreement (federation) between the two TSs (with the user consent), but otherwise, and in general, multiple unlinkable identities will be a matter of course. The problem of trust is related to the authority that vouches for the user identity, e.g. a certification authority. The experience on the Internet shows that a single global PKI is a dream, so there will be many (and in general uncontrollable) certification authorities. Another approach is a distributed "web of trust", without certification authorities, as in PGP. Anyway, different TSs will trust different authorities, and this is a problem for a user that needs to access multiple TSs. The problem of federation and trust have been recognized by the web community and resulted in the SAML effort at OASIS. SAML can be a good candidate technology to federate multiple spaces.

- *Authorization*: as authorization relies on authentication, authentication problems have a great impact on authorization. Moreover, other specific problems may arise, for example about different policies in different TSs.

- *Accounting and logging*: if, in order to detect that a user did something bad, logs from different TSs need to be correlated, we must be sure that we are talking about the same user (see authentication problems), and we must trust each TS owner to log correctly. Indeed, each TS owner is a sort of "log authority", and the problem we face is the problem of trust in a world with many unrelated authorities.

The problem of different policies and/or different accounting and logging principles in independent triple spaces could be attacked by the idea of "mediators" as described

in the WSMO concept. There is no difference between the problem of mediating between applications and of mediating between different security systems. Indeed, the security field was the first to understand the issue of "mediators" in practical terms, when e-provisioning systems were invented. Such systems "mediate" between heterogeneous business systems in order to provision automatically the resources required to achieve a specific goal (a typical example is when a new employee joins a company and the system provisions all necessary resources at the press of a button: the phone, the e-mail connection, the Internet connection, the access rights to libraries, the security card to enter the office, the physical workstation, the corporate directory, subscriptions to information sources, etc). Again, these "policy mediators" are a component of a "semantic security" world, which is out of scope in TripCom project but is useful to be kept in mind as an inspiration for future developments.

2.3 Information trustworthiness

The TS owner is like the administrator of a web message board: he must set policies and arrange things so that users register and authenticate themselves if needed, access only data they are allowed to access, and so on, but usually does not mind what users say. Indeed, the TS owner may want to adopt policies aimed at guaranteeing some degree of trustworthiness to data published in the space he manages, but his major role is deciding who can do what in his TS, and not deciding what is "right" (true, trustworthy) and what is "wrong" (false, untrustworthy) in data published by users.

Dealing with the uncertainty about the information trustworthiness is another side of the *trust* problem, that arises because, in a single TS as well as in the WTS, there is not a central "authority" that states what is true and what is false (note that now we are not talking about security policies, but about information published in the TS). Users have partial knowledge of what is around them, and with this knowledge they must decide whether they want to interact (and to what extent) with other entities, and whether they want to trust the information other entities publish. This is distinct from authentication, because knowing someone's identity is usually not enough in order to decide whether to trust him.

2.3.1 Reputation

Trust relationships can be built using *reputation*: users can express their opinion (in some way) about other users and/or about statements, basing their judgement on their previous experience. Each user can use this information in conjunction with his personal experience to build his own, subjective picture: indeed, each user can compute his *subjective trust values* using different weights for opinions expressed by different entities, according to his own experience. Different trust values about the same statement are not a contradiction that should be removed: for example, some users may agree with a statement like "this is a good restaurant" while others may disagree, and there is nothing wrong in such a situation (moreover, each user may take into particular account opinions by users with the same likings as him). *Global trust values* could also be defined: a global trust value can be a sort of mean value of all opinions (not biased by the experience of the particular user who computes the value) and it may be computed and used by the infrastructure, or even used by users who do not have enough experience in order to compute a subjective value.

Subjects (users) can express their opinions about objects (resources, statements) or about other subjects; therefore we can have both resource reputation and user reputation:

- *resource reputation* is a way each user has to settle how much a statement can be taken into account, using the opinions that other users expressed about it;
- *user reputation* is a way to enforce some sort of "social accountability" in a world where there is no global authority able to do it; reputation in general does not need to be linked to real world identities, and is compatible with pseudonymity.

The mechanism used to distribute trust information must be flexible and general–purpose, applicable to different entities in the system (both subjects and objects) and to different goals (e.g., trust in a user as an "author" vs. trust in a user as a "recommender"). While computing global trust values can be an infrastructure matter, subjective trust values are an application matter: the infrastructure should provide a mechanism to distribute relevant information, while the problem of how to use this information in order to compute subjective trust values is left to each application or service according to its needs.

2.3.2 Trust negotiation

Finally, apart from using reputation to make a decision about trusting some data, users and services may establish a sort of trust channel — they may negotiate a set of service parameters and guarantees (also known as Service-Level Agreements, SLAs), which may improve the trust evaluations.

For example, a source of data may be authenticated as a legal entity in U.S.A., but this may not suffice to trust their data. However, if that source also agrees to penalties in case their data is erroneous (not truthful), or if the price of error is negligible, the user may decide to trust the data.

3 Security of the infrastructure

3.1 Security services and systems

This section briefly introduces general security services, in order to provide and explain a common terminology for the subsequent sections.

3.1.1 Authentication

Authentication is the process of verifying a claimed identity, i.e., to verify that someone (user, machine, etc.) is really who he/she/it claims to be. This is usually accomplished using one or more *authentication factors* (credentials), which can be divided into three groups:

- something the user *knows*, i.e., some secret information (e.g., password, PIN);
- something the user *has*, i.e., some physical object (e.g., smart-card, badge);
- something the user *is*, i.e., some biometric data (e.g., fingerprint, iris pattern).

Many authentication mechanisms and protocols exist; anyway, it is useful to divide them into two large groups, or patterns:

- *Direct authentication*, where a user presents its credential in order to establish the identity required in the interaction. Direct authentication methods require that the authenticating party provides all facets of an authentication service, without relying on some other party (e.g., a database system supports direct authentication by managing credentials in internal system tables). Individual web sites like e-commerce sites often use direct authentication methods; most web sites provide their own registration, user management and authentication systems.
- *Indirect authentication*, where a trusted third party is involved in the authentication process, and vouches for the user identity. Indirect authentication methods can be further divided into two groups:
  - *Brokered authentication*, where the trusted third party is used to broker authentication service, being directly involved in the authentication process between the two parties. For large, distributed infrastructures like enterprise systems, brokered authentication is considered to be the method of choice (Kerberos [55] is a good example). On the web, various projects for brokered authentication have been initiated, like the famous Microsoft "Passport" system. Although technically superior, brokered authentication systems on the web ran into massive privacy problems since they can embody the "big brother" concept.
  - *Offline authentication*, where the trusted third party vouches for the user identity but is usually not involved in the authentication process between the two parties, which can take autonomous decisions without contacting the authority. An example of offline authentication is given by PKIs: a service does not need to contact the certification authority every time a

user presents a certificate; anyway the service accepts the risk of using stale authentication data, and therefore taking wrong decisions (e.g., because the certificate has been revoked but the service uses an old version of the certificate revocation list). Offline authentication can be considered more "publication oriented" than brokered authentication.

3.1.2 Access control

Once the user has been authenticated, the system needs to give him correct access rights: it evaluates access requests to resources and, basing on some access rules, it determines whether they must be granted or denied.

Access control models are traditionally divided into two groups:

- *Discretionary Access Control* (DAC) models, where individual users can grant and revoke at their discretion access privileges on objects under their control. In such systems there is a concept of object ownership: each object has an owner, i.e. the user who controls its access privileges.
- *Mandatory Access Control* (MAC) models, where there is an access control policy determined by the system, and not by individual users. Such models are used for example in military systems, basing on object classification (e.g., Top Secret, Confidential) and user security clearance.

A newer model is *Role Based Access Control* (RBAC) [36]: in RBAC systems, access decisions are based on the roles that individual users have as part of an organization (e.g., doctor, manager, etc.). Users take on assigned roles, access rights are defined using roles, and access to resources is restricted to individuals authorized to assume the associated role.

The elementary abstraction used to represent access rights is the *Access Control Matrix*. Having a set of objects (resources to be protected, e.g., files in a file system), a set of subjects (entities that access resources, e.g., users and processes), and a set of access rights (e.g., read, write, etc.), we can represent access rights with a matrix where each subject is a row, each object is a column, and each cell represents the access rights that subject has on that object. *Access Control Lists* (ACL) and *capability lists* can be thought as representations of the matrix by columns and by rows, respectively.

3.1.3 Anonymity and pseudonymity

Knowing the user's identity is a requirement in many applications; in some cases, however, the opposite can be true, i.e., the user must be certain that his identity cannot be disclosed (anonymity). We can identify different kinds of anonymity:

- *sender anonymity*: the recipient of a message cannot know the sender's identity;
- *recipient anonymity*: the sender of a message cannot know the recipient's identity;
- *communication anonymity*: a third party cannot know the identities of the parties involved in a communication.

A weaker form of anonymity is often called *pseudonymity*: this is the case when each user has an identifier (pseudonym), but this identifier is not bound to a "real world" identity, so that it cannot be used to identify the real person behind it. Moreover, the user can usually change his pseudonym whenever he wants, or even use more than one pseudonym at the same time. Thus, the user is fundamentally anonymous, but the pseudonym allows to link subsequent actions made by the same user (i.e.: "I don't know who you are, but I know you are the same as yesterday").

Anonymity is usually achieved using some sort of mediators, often in conjunction with cryptographic mechanisms. MIX networks [25] are a classical example.

3.2 Internet and web security

3.2.1 Indirect authentication solutions

A single TS could use direct as well as indirect authentication; anyway in the WTS the availability of indirect authentication services is highly desirable.

Kerberos [55] is a brokered authentication solution, which allows users to authenticate to different services in a network using *single sign-on* and without giving user passwords to services. Kerberos is a centralized solution, based on a *key distribution centre* (KDC) that acts as a trusted third party to users and services. The KDC must be on-line, because it must issue to users *tickets* vouching for their identity. Kerberos is quite common on Unix systems, and has been adopted by Microsoft starting from Windows 2000. In order to use Kerberos authentication, services must be "kerberized"; since Kerberos API is not standardized this is often done through GSSAPI. Kerberos usage is less common in web applications, but HTTP authentication using Kerberos is supported by major web browsers and servers via SPNEGO [71].

PKIs (Publick Key Infrastructures) can be classified as offline authentication: upon enrolment, users get a certificate from a certification authority, vouching for the relationship between their identity and their public key. The certification authority acts as a trusted third party, and if the service trusts the certification authority it recognizes the user. Unlike Kerberos tickets, which are targeted to a particular service and have a very limited time validity (usually a few hours), X.509 certificates have more general purposes (e.g. "client authentication") and longer validity (usually some years); therefore, there is no need to involve the certification authority in the authentication process between the client and the server[1]. PKI usage is very common for server authentication on the Web, as X.509 certificates are used to authenticate HTTPS servers (client authentication is less widespread).

3.2.2 Network and transport layer communication security

IPsec and TLS/SSL are security protocols used at network and transport layer respectively. They both protect information only during network transit: they complete their job when data is delivered to destination host/gateway (IPsec) or application (TLS/SSL). In TripCom, they can be used as standard solutions to secure communi-

[1] Anyway, the certification authority must be contacted from time to time in order to get an up-to-date version of the *Certificate Revocation List* (CRL), which lists certificates that have not yet expired but have been revoked.

cation channels between different TS infrastructure components and between the TS and agents.

IPsec (Internet Protocol Security) [43] is a security architecture for the IP protocol. It defines three protocols (AH — Authentication Header, ESP — Encapsulating Security Payload, IKE — Internet Key Exchange) and some data structures used to handle secure communication properties. IPsec provides confidentiality, data origin authentication, connectionless integrity, anti-replay service, access control. A key concept in IPsec is the *security association*, i.e. an agreement between the two communicating parties that specifies which cryptographic algorithms and keys will be used in the communication. Security associations are created and managed using IKE, and are used by AH and ESP. IPsec protects traffic at network layer (IP packets), and is mainly used in virtual private networks (VPNs). It can be used to protect traffic between hosts or between entire subnets (tunnelling traffic through security gateways).

TLS (Transport Layer Security) [31] secures the TCP communication channel between two applications, providing confidentiality, integrity, and authentication. It is an IETF standard and an evolution of the SSL protocol (which has never been standardized). Unlike IPsec, TLS is a client–server protocol and is used directly by applications. TLS is used with many application layer protocols, and in particular with HTTP and e-mail transfer protocols (SMTP, POP, IMAP).

3.2.3 Group communication security

Group communication (i.e., one–to–many and many–to–many communications) may have the same basic security needs as one–to–one communication, i.e. confidentiality, authentication, integrity, etc. Anyway, group communication has peculiar security problems.

Regarding authentication, in group communication we must distinguish between *source authentication* and *group authentication*: the first is the "usual" authentication, i.e. assuring the identity of the information source, while the second means just the assurance that the source is one of the group members. Usual message authentication codes (MACs), as used in TLS and IPsec, give group authentication (since the key is known by all group members, they are all able to compute the MAC); source authentication can be achieved using digital signatures but, as this is computationally very expensive, other solutions exist (e.g. TESLA [59]).

Groups are dynamic entities: members can join and leave, and membership can be revoked. Usually, membership of a secure group is given by knowing some secret information (i.e., a key or a set of keys) that allows to participate in group information exchange. The basic problem of group communication security is therefore *key management*. A "group key", known by all members, is usually used to encrypt group data, and can be established by a "controller" (group owner) that gives it to each member using one–to–one secure channels. When a new member is added, the group key must be changed in order to prevent the new member from accessing past communications to the group (*backward access control*); this is easy to do, as the new key can be sent to the group encrypted with the old key using a "group" message, and to the new member using a single message on a one–to–one channel. It is much more difficult to assure *forward access control*, i.e. preventing a revoked member to access subsequent data: the group key must be changed so that all members but the revoked one get it, usually without using a single channel for each remaining member (which

would be very expensive if the group is large). Many group rekeying algorithms have been proposed, both stateful (requiring members to receive all rekeying messages and keep an updated state, e.g. LKH [70]) and stateless (requiring members to maintain only additional keys established during registration phase, e.g. Subset Difference [53]). These schemes are used in multicast and broadcast communications, with or without the need for a return channel. Since publication on a shared space can be somewhat similar to broadcasting, these solutions can be interesting for TripCom in order to secure one–to–many and many–to–many communications.

3.2.4 E-mail message security

E-mail messages can be protected end–to–end using S/MIME or OpenPGP (both are IETF standards).

S/MIME [34] is a security extension to MIME format, and is applicable to any use of MIME. It can provide confidentiality, authentication, integrity and non repudiation of messages.

OpenPGP [22] format comes from PGP software, and was later standardized by the IETF. OpenPGP goals are basically the same as S/MIME, anyway there is big difference in certificates. Indeed, OpenPGP uses its own certificates, rather than usual X.509 certificates, in order to realize a *"web of trust"*. OpenPGP certificates can include many signatures (and not only the signature of the certification authority that issued the certificate), so that users themselves can sign other users' certificates. Therefore, certificates are organized in a graph (web), rather than a tree as with usual PKIs. OpenPGP can be a good solution in "collaborative" scenarios where central entities like certification authorities are not appropriate.

3.2.5 Web Services security

The key benefit of Web Services architecture is the ability to deliver integrated, interoperable solutions via standardized message exchange. Part of the Web Service standards therefore is a comprehensive security model ensuring *integrity*, *confidentiality* and *security* of communication and services.

SOAP plays a major role in Web Service land. Unfortunately, its meaning is widely misunderstood. *SOAP is a messaging architecture*. It defines the XML based information that enables the exchange of typed and structured information between peers in a distributed, decentralized environment. SOAP is a stateless, one-way message exchange paradigm — more complex interaction patterns are defined on top of SOAP. It has its beginnings in RPC over the web, but since v1.2, SOAP doesn't mean "Simple Object Access Protocol" anymore, it now is an acronym that stands for itself [39].

Web Service security is not meant to replace any existing security technology (e.g. SSL/TLS, IPsec), neither at the transport level nor at the network level. Instead, WS-Security augments and federates existing security infrastructures and provides a unified model for application programmers and network administrators. E.g. WS-Security uses Kerberos or X.509 for authentication, XML-Encryption for encryption and XML Signatures for signing messages. It is a standard that describes a framework how to reuse existing security mechanisms in a SOAP message, independently from transport technology. It therefore defines appropriate SOAP Header elements to describe security relevant information like the algorithm used for encryption, the

kind of key used and the resulting signature. The actual format of the signature is left to the specification of the security mechanism itself, WS-Security only describes how to embed this technology into a SOAP message. You can see WS-Security as a specification for a XML based metadata container for SOAP security.

SOAP is architected to allow messages to go through one or more SOAP intermediary nodes before they reach their final destination. Intermediaries may access the header and body of a message, possibly changing their contents. This means that network or transport layer security needs to be terminated at each intermediary. As a result, security on the transport layer alone is not sufficient to ensure secure communication via SOAP. Additionally end–to–end security mechanisms which respect the SOAP processing model have to be defined on the message layer.

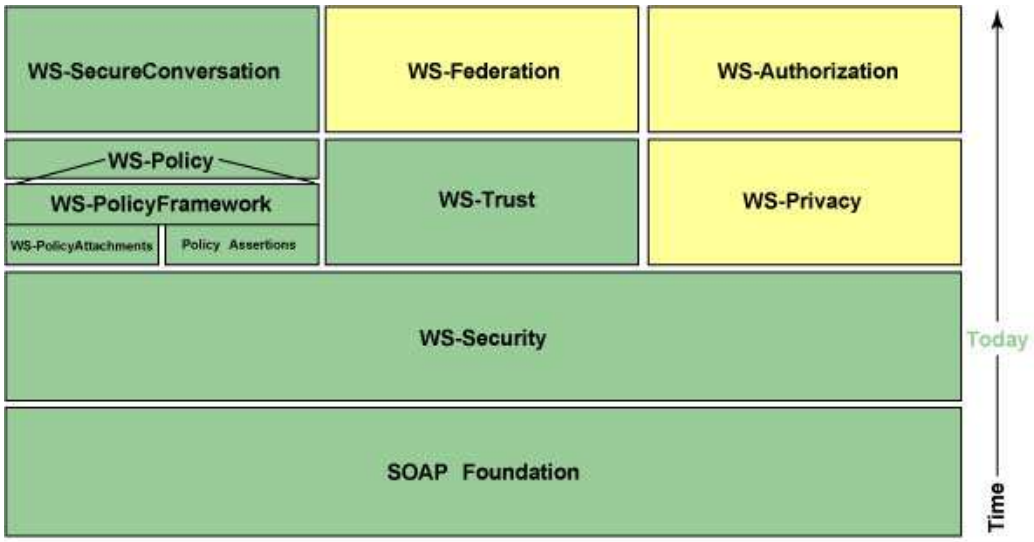

Figure 3.1: Web services security specifications.

Web Services Security Standards

WS-Security (SOAP Message Security 1.0) WS-Security [52] provides a common syntax and processing model for carrying security information in SOAP envelopes, enabling the integration of existing enterprise and internet security mechanisms. Just as outlined by figure 3.1, WS-Security (and SOAP) is the base standard for all further Web Service Security standards. Major building blocks are also XML Encryption [40] and XML Signatures [14].

WS-Security describes enhancements to SOAP messaging to provide quality of protection through *message integrity* and *message confidentiality*. As well, this specification defines how to attach and include security tokens within SOAP messages. Finally, a mechanism is provided for specifying binary encoded security tokens (e.g. X.509 certificates). These mechanisms can be used independently or in combination to accommodate a wide variety of security models and encryption technologies.

WS-Security provides a general–purpose mechanism for associating *security tokens* with messages. No specific type of security token is required by WS-Security. It is designed to be extensible (e.g. support multiple security token formats). For example, a requester might provide proof of identity and proof that they have a particular business certification.

Message integrity is provided by leveraging XML Signature in conjunction with security tokens (which may contain or imply key data) to ensure that messages are transmitted without modifications. The integrity mechanisms are designed to support multiple signatures, potentially by multiple actors, and to be extensible to support additional signature formats. The signatures may reference (i.e. point to) a security token.

Similarly, *message confidentiality* is provided by leveraging XML Encryption in conjunction with security tokens to keep portions of SOAP messages confidential. The encryption mechanisms are designed to support additional encryption technologies, processes, and operations by multiple actors. The encryption may also reference a security token.

Finally, WS-Security describes a mechanism for encoding binary security tokens. Specifically, the specification describes how to encode X.509 certificates and Kerberos tickets as well as how to include opaque encrypted keys. It also includes extensibility mechanisms that can be used to further describe the characteristics of the security tokens that are included with a message.

WS-Trust WS-Trust [7] describes the model for establishing both direct and brokered *trust relationships* (including third parties and intermediaries). This specification describes how existing direct trust relationships may be used as the basis for brokering trust through the creation of security token issuance services. These security token issuance services build on WS-Security to transfer the requisite security tokens in a manner that ensures the integrity and confidentiality of those tokens.

The specification then describes how several existing trust mechanisms may be used in conjunction with this trust model. Finally, the trust model explicitly allows for, but will not mandate, delegation and impersonation by principals.

WS-Privacy (not yet published, but planned according to [4]) Organizations creating, managing, and using Web services will often need to state their privacy policies and require that incoming requests make claims about the senders' adherence to these policies.

By using a combination of WS-Policy, WS-Security, and WS-Trust, organizations can state and indicate conformance to stated privacy policies. This specification describes a model for how a privacy language may be *embedded into WS-Policy descriptions* and how WS-Security may be used to associate privacy claims with a message. Finally, this specification describes how WS-Trust mechanisms can be used to evaluate these privacy claims for both user preferences and organizational practice claims.

WS-Federation WS-Federation [5] is a family of three specifications: WS-Federation, WS-Federation Active Client and WS-Federation Passive Client. These specifications define *how to construct federated trust scenarios* using the WS-Security, WS-Policy, WS-Trust, and WS-SecureConversation specifications. For example, they describe how to federate Kerberos and PKI infrastructures. As well, a trust policy is introduced to indicate and constrain and identify the type of trust that is being brokered. Also, mechanisms for managing trust relationships are defined.

In short, WS-Federation defines mechanisms for brokering and federating trust, identity and claims. Federation is the overall term for a set of distinct, heterogeneous

enterprises that want to provide easy–to–use, single sign-on identity model to their users.

WS-Authorization (not yet published, but planned according to [4]) This specification describes how access policies for a Web service are specified and managed. In particular it describes how claims may be specified within security tokens and how these claims are interpreted at the endpoint.

This specification is designed to be flexible and extensible with respect to both authorization format and authorization language. This enables the widest range of scenarios and ensures the long–term viability of the security framework.

This specification again is build on top of SOAP Message Security and defines primitives and extensions for security token exchange which facilitates authorization within federated administrative domains.

WS-SecureConversation While WS-Security enables confidentiality and integrity of a single SOAP message, typical Web service interactions involve multiple messages to be exchanged. Rather than encrypting and signing each message of an interaction, WS-SecureConversation [6] *facilitates the exchange of long–lived security credentials* between the communication partners at the beginning of a conversation. These credentials (i.e. Security Context Tokens, SCT) are acquired by the service requester and the service provider at a Security Token Service (STS) and can then be used as a shared secret (e.g. a symmetric session key) during the actual message exchange.

WS-Policy WS-Policy [8] describes a grammar that enables senders and receivers *to specify their requirements and capabilities* and compose them as combinations of domain assertions, without placing any limit on the types of requirements and capabilities that may be described. Furthermore it defines a way how policy compatibility can be determined through the mechanism of policy intersection. WS-Policy Attachment, which is also part of the WS-Policy framework, describes mechanisms for *attaching service policies to Web service endpoints.*

3.3 Security and trust in the Semantic Web

3.3.1 Web security and trust on the Semantic Web

While the approaches to security and trust outlined thus far in this chapter can complement or exist orthogonally to the Semantic Web, there are some aspects of security and trust which can be reconsidered in the context of the Semantic Web paradigm:

- the use of vocabularies defined in RDF Schema or OWL to express security or trust information;
- the ability to apply logic-based reasoning over such semantically enriched information in order to draw conclusions regarding the security of some data or the trustfulness of some agent;
- the particular characteristics and semantics of the RDF data model changing the meaning and approach to data encryption.

Due to these aspects, we briefly draw together some work in the Semantic Web field relating to security and trust.

3.3.2 Semantic Web vocabularies for security and trust

The Semantic Web Publishing Vocabulary (SWP) [23] is intended for expressing provenance meta-information about RDF graphs. SWP provides terms to indicate whether a graph is asserted or quoted and to attach digital signatures to it. The WOT, or Web Of Trust, schema (`http://xmlns.com/wot/0.1/`) is designed to facilitate the use of Public Key Cryptography tools such as PGP or GPG to sign RDF documents and document these signatures.

In order to protect digital content, rights about its use must be expressed in some standard-respecting way. A Semantic Digital Rights Management System (`http://rhizomik.net/semdrms/`) has been implemented using a core copyright ontology together with ontology models of the MPEG-21 and OPRL standards.

For a distributed setting, Rein (`http://groups.csail.mit.edu/dig/Rein/`) realises a policy framework where different users can express policies in their own (RDF or OWL) taxonomies and uses rules to enable an user to access and reason over others policies. It also makes use of the policy specification language Rei (`http://rei.umbc.edu/`). To quote from their website: "Rei is a policy language based in OWL-Lite that allows policies to be specified as constraints over allowable and obligated actions on resources in the environment. Rei also includes logic-like variables giving it the flexibility to specify relations like role value maps that are not directly possible in OWL. Rei includes meta policy specifications for conflict resolution, speech acts for remote policy management and policy analysis specifications like what-if analysis and use-case management making it a suitable candidate for adaptable security in the environments under consideration".

SWAD has made public a deliverable on vocabularies and an architecture for implementing trust in the Semantic Web (`http://www.w3.org/2001/sw/Europe/reports/trust/11.2/d11.2_trust_vocabularies.html`). They propose a trust statement which expresses that an Agent A (human or machine) can trust another Agent B for some Action X, within some time bound T1 to T2 within constraints (or context) Y, to a trust level defined with some trust metric defined by some URI `http://example.org/trustMetric`. The TPL - Trust Policy Language is used in TriQL.P (`http://sites.wiwiss.fu-berlin.de/suhl/bizer/TriQLP/browser/index.htm`) to define trust policies. A trust policy in TPL consists of a policy name and description, graph patterns, constraints and explanation templates.

3.3.3 Semantic Web approaches to security and trust

PeerTrust [54] is a current project about automated trust negotiation for peers on the Semantic Web. To quote from their website: "The PeerTrust project is investigating trust negotiation in Semantic Web and P2P environments. Within the program, digital credentials can be signed XML or RDF statements that express peer properties, and policies are expressed as logic programs that tie resource access to required credentials. The ability to refer to peers, to credentials, or to other resources in PeerTrust logic programs lets us express the iterative exchange of credentials during a trust negotiation process".

Expressing and reasoning over policies is also part of the European Network of Excellence REWERSE. They have produced a good overview of requirements and current research issues in Semantic Web Policies [18].

Policies are also the focus of an initiative to build the "Policy Aware Web" (`http://www.policyawareweb.org/`), a rule-based access control for the Web. Previous work in this area has been done in KaOS [69], which has produced a policy framework and toolset based on a DAML description-logic-based ontology that allows the specification, management, conflict resolution and enforcement of policies within an agent, grid or web service context.

SWAD-Europe has an activity in trust research which has made some draft reports and organizes an annual iTrust conference. A set of scenarios for using RDF in support of Trust and Access Control is given in (`http://www.ninebynine.org/SWAD-E/Trust-scenarios.html`). It includes role-based access control and a medical record access scenario.

Most Semantic Web work at present is not part of a larger initiative but is done in individual research. A typical Web of Trust architecture based around "believes in" statements and 'trust' between agents is described in [49]. Reputation is also extendable with recommendations, responsibility and trust among peers (`http://www.w3.org/2001/sw/Europe/events/foaf-galway/papers/fp/redefining_web_of_trust/`). Another research work [41] presents an infrastructure that complements existing security features like Public Key Infrastructure (PKI) and Role Based Access Control with a distributed trust management system. Content-based and context-based trust mechanisms [16] are proposed in addition to reputation-based. Even the most recent work continues to focus on logical rules for trust policies (`http://www.l3s.de/~olmedilla/events/2006/SWPW06/programme/paper_14.pdf`).

An interesting paper (`http://www.w3.org/2002/03/key-free-trust`) introduces an approach towards trusted semantic web applications that do not require the existence of an complex public key infrastructure. Rather, it uses dense meshes of interrelated statements for trust evaluation instead of the usual reputation-based approach.

Finally, (`http://www.ninebynine.org/SWAD-E/Security-formats.html`) is a good reference for current security and trust standards and discusses how they might be integrated in a Semantic Web trust and policy framework.

3.3.4 Handling the RDF data model

One new aspect of the RDF model is the varying granularities with which one works with the data. Groups of statements are often related, raising the requirement of signing individual fragments on an RDF graph [68].

Efficient algorithms are needed for computing the digest of a Resource Description Framework (RDF) graph which may be used to assign unique content-dependent identifiers and for use in digital signatures, allowing a recipient to verify that RDF was generated by a particular individual and/or has not been altered in transit [12]. Jeremy Carroll presents an alternative approach to solve this problem of creating and verifying a digital signature for a RDF graph [24].

Most RDF based approaches to signing and encrypting RDF data use W3C XML Signature and XML Encryption with the RDF/XML serialization. A Cryptography Ontology (Crypto) specifies a set of RDF classes and properties to use to specify encryption-related information in an RDF instance. Crypto is derived in part from

the W3C "XML Encryption Syntax and Processing" and "XML-Signature Syntax and Processing" specifications (`http://lists.w3.org/Archives/Public/www-rdf-interest/2003Dec/att-0140/crypto-draft-20031224.html`). Encrypting RDF again needs to consider the various possible granularities. Partial RDF Encryption (`http://www.vis.uni-stuttgart.de/~giereth/publications/eswc05-giereth-paper-publishing.pdf`) allows to encrypt selected fragments of a RDF graph for selected addressees while all other parts remain publicly accessible.

3.4 Storage security

The TripCom storage architecture is the lowest level in the TripCom architecture, handling the actual physical storage of data. The storage layer is composed of data sources (e.g. RDF-stores, relational databases, XML files) and *storage middleware*[2] to manage the access to these data sources and do any necessary transformations.

Due to the heterogeneity of the data sources, the security model for data sources will be fairly simple and will allow only two levels of permissions: read and full control (read+write). To accomplish this, the storage middleware must be able to authenticate its users (cf. Section 3.1.1) and the communication between the higher layers and the storage middleware (and also the data sources) must be confidential (over a local/private network, or encrypted, cf. Section 3.2.2).

There are two important additional points worth mentioning:

- It is **not** a requirement that an administrator of a data source and/or an instance of the storage middleware would have limited access to the data in the storage. Therefore if any data should be confidential from the administrator, it must be encrypted before it is handed off to the storage layer.
- Because the administrator of a data source and/or an instance of the storage middleware has full access, trust towards them (i.e. trust towards a specific storage middleware instance) must be an explicit consideration, and steps may need to be taken (e.g. using digital signatures) to assure the authenticity of data in the storage.

This simple storage security model is sufficient to support any more complex security and trust models described in other parts of this document, and those will be implemented in the TripCom architecture layers above the Storage.

3.5 Tuple space security

The tuplespace paradigm foresees communication between clients through a shared virtual dataspace known as a tuplespace. Given the differences between tuplespace communication and other forms of network communication it is important to consider how issues of trust and security can be solved specifically for tuplespaces. For example, generally any client can leave any data in a space and any other client may, if it seeks data matching the data that was left, retrieve that data from the space. Both the data provider and consumer in classical tuplespace models would remain anonymous

[2]The term *storage middleware* is not yet finalized.

to each other and even also to the space. In an open system like the Internet, we can not assume that all clients acting in the space are trusted or safe. Hence open tuplespace systems must also implement means to protect both clients and itself from other, possibly malicious, clients who may be posting false data to the space or bringing down the space (e.g. by flooding the system with requests). In this chapter we introduce the general security framework for shared space co-ordination presented in [20], extended with observations from other implementations of security in tuplespace systems.

3.5.1 General Security Architecture for Tuplespace

A security framework needs to consider client authentication and authorization as well as to use encryption to protect communication in an open environment. It can be built upon a Trusted Computing base (TCB) which contains the code and data for secure system operation, e.g. authentication procedures, access rights and encryption keys. The TCB must be protected as it is fundamentally the weak point of any system, i.e. if the TCB is compromised, the security of the entire system is compromised. Hence, the first fundamental component in a secure tuplespace system can be called the reference monitor, which should check that each operation on the space by a certain agent is permitted according to the valid security policies. Another functionality of the reference monitor may be to ensure that aliasing can not be used to bypass the coordination language primitives when clients share references to data objects placed in the space. The reference monitor may be implemented as a proxy for the space [50] or as part of the space [21].

A reference monitor will need to refer to security policies, which is a set of rules defining how interactions with the space are regulated. Connected to this is a means to identify clients (authentication) and to verify what is allowed to them in the system (authorization). Issues that must be considered in an Internet co-ordination context include:

- the number of active clients may be very large, making authentication difficult;
- network requests may come from unknown sites, meaning the system must be able to make a decision on accepting the request or not;
- Linda, the coordination language of tuplespaces, uses associative access for data retrieval and the security infrastructure should not undo this benefit.

3.5.2 Authentication

A client must be identified in order that the appropriate access rights can be given to it. However, there can be types of interaction where the identity of the client should not be revealed and hence the client should also be able to prove to the system that it is authorized to do certain interactions without identifying itself. The data that a client provides to authenticate itself is called its credentials. [20] proposes that, as each client may use the space over its own protocol, that a "protocol authentication process" (PAP) is used which is a designated client (external) or system process (internal) which handles authentication requests for a particular protocol. The client presents its credentials to the PAP and if successful, receives an authentication token. The client then provides the authentication token to the system when interacting with the

tuplespace. An example of an implementation of this approach is Kerberos [64]. The token can correspond to a set of access rights, or can act as a cryptographic key which is used to decrypt subsequent messages [21]. The PAP must also be authenticable, so that a malicious client can not masquerade as a false PAP. Finally, tuples retrieved from a space should also be authenticatable as being truly from a certain client or at least being from some client which is authorized to provide data. [20] provides a cryptographic approach to authentication protocols.

3.5.3 Authorization

In the tuplespace paradigm, a space or an individual tuple can be accessed in principle by any agent. Hence one issue for tuplespace authorization is the granularity of access privileges: to a space, partition of a space, a tuple or even to fields of a tuple. Also, in an open system scenario, one must consider the possibility that access rights are dynamic, i.e. that they can change over time. Finally, a remaining issue is that of who is able to grant and revoke access rights in the system. In KLAIM [56], for example, tuple creators may determine the rights of other clients to their tuples.

Access rights information must be protected in the system so that clients can not alter it. A typical approach is through typing, e.g. a system implemented in Java can use the language's safe typing constructs [56]. In an open distributed system this may not be enough as rights information may be transferred over the network or stored in an untrusted environment, so encryption is also needed [65]. In KLAIM, access rights are implemented as first class objects, i.e. they can be exchanged between clients as tuple data through the space. As they are typed, security becomes an issue of type correctness.

Both capabilities and access control lists can be modified to apply to the tuplespace model. A capability for example could be described as —(PurchaseOrder,out),(Invoice,in)— (the client may only emit purchase orders to the space and may only destructively remove invoices). In an access control list one could state Invoice := —(Seller,out),(Buyer,in)—; PurchaseOrder := —(Buyer,out),(Seller,rd)—. Here we state that a client authenticated as a Seller may emit invoices to the space and clients authenticated as Buyers may destructively retrive them. Likewise, Buyers emit PurchaseOrders which Sellers can non-destructively retrieve. The role-based access control model (RBAC) may be a suitable choice for ACLs in tuplespace systems.

In Law-Governed Linda [51], controllers exist for each client and mediate between it and the tuplespace, applying some "law" which specifies how communication should take place (i.e. specifies which operations can be permitted with data matching some given templates). Each law applies to a group, and clients can belong to different groups or change groups. Controllers will organise their laws dynamically according to the agent's current groups and hence maintain complete knowledge over the agent's access rights.

In TuCSoN [27], the tuplespace is made up of shared spaces called "tuple centers" which may each contain its own coordination rules. These rules are expressed in the space as reactions of the form (Op,R) where Op is some operation and R is the specified reaction. Hence agents may be moved between different tuple centers (according to their access rights) with different defined reactions (e.g. a center may contain the reaction (in,null) meaning do nothing when a client tries a destructive read).

3.5.4 Encryption

Both security and application data may require encryption to protect malicious access and safe network communication. One aspect of tuplespace systems which impacts on encryption is the possible ambiguity of who is the source and who is the target of a given tuple communication. [20] suggests that the credentials need to be part of the tuple — i.e. the tuple with a purchase order intended for a specific seller contains a certificate verifying that a particular (trusted) buyer has emitted the tuple and the purchase order data encrypted with a key which the seller has (and hence the seller can know he is authorized to take the data). SecOS [21] takes this idea, associating keys with tuple data. A client must have a matching key in order to be able to access the corresponding tuple data.

However this is insufficient to prevent a malicious client from taking a tuple it has rights to, re-encrypting the data according to another key and outputting the tuple so that another (unauthorized) client has authorized access to the data (by having the other key). As a result, one-way hash functions could ensure that data in a tuple has not been modified [63]. Further issues arising from distributed systems and applicable to tuplespace systems can be found in [9].

3.5.5 In a world of Triple Spaces

Given the intention of Triple Space to be Web scalable, it seems highly likely that rather than there be a single global Triple Space that there will be a world of Triple Spaces analogous to the multiple of web servers which form the World Wide Web.

Each space will have its own reference monitor and security policies. The system must be able to act with partial knowledge of security and trust policies. For example, this could mean the distribution of security and trust policies across reference monitors according to which data each space holds as well as means for reference monitors to access policies at other spaces when unable to authenticate a client or authorize an action on the basis of the policies that it has. One possible approach here is delegation [45]. TuCSoN is an example of a tuplespace system taking this approach [27]. PKIs typically delegate authentication requests and SPKI [35] is an example of a framework which delegates authorizations. This also leads to considering applying security policies not only to clients and tuples but also to spaces.

3.6 Middleware security

3.6.1 Introduction

Middleware security encompasses a wide range of potential considerations, from standardized mechanisms embodied in classic middleware frameworks like CORBA, .NET, J2EE to complex distributed security solutions. Considering the TripCom architecture, the middleware represents a layer which supports the Tuplespace core components and ties these components to communication and service components. This means that we limit the discussion of middleware security to the case where middleware plays the role of an underlying framework.

The categorization of middleware security can be efficiently based on the security meta-model of CORBA [30, 38]. The CORBA security meta-model deals with

interacting objects in a general form, e.g. it is an implementation neutral model for middleware security.

It should be noted that the CORBA specification goes far beyond the definition of a metamodel. The CORBA Security service protocol defines the core security facilities and interfaces required to ensure a reasonable level of security of a CORBA-compliant system as a whole. However, since we foresee TripCom systems to leverage on WebService compliant protocols, we concentrate on the categorization of the security services, rather than on the specific CORBA interfaces.

Middleware security is thus considered to be a hub that relates the outside (WebService (WS)) security protocols (like WS-Security, WS-Policy, WS-Trust, WS-Privacy, WS-SecureConversation, WS-Federation, WS-Authorization, etc.) into mechanisms embodied by the inner components of the system.

3.6.2 Categorization of middleware security

The security meta model of CORBA defines four security related aspects that expand into six features:

- confidentiality aspect expands into the features "Identification" and "Authentication";
- integrity aspect expands into the features "Authorization and Access control";
- accountability aspect expands into the feature "Security Audit";
- availability aspect expands into the features "Secure Communication" and "Security Administration".

These aspects characterize the interaction of objects at run-time. The implementation of a middleware infrastructure requires also the definition of the "participation" level of each component: the "participation" level defines the way a specific component participates in the realization of the security aspects above. CORBA defines three levels of participation [17]:

- security unaware: the application relies completely on the middleware for security;
- security policy controlling: the application directs the middleware to use specific, predefined policies, but relies on the middleware for enforcement;
- security policy enforcing: the application directs middleware to use specific policies and actively examines security context information to enforce policy as well as relying on middleware enforcement.

Translated into the context of the Tripcom architecture, the "participation" level is of central importance: relying on middleware security alone leads to a static security system, e.g. a system where security policies can be formulated on the base of predefined categorizations like roles, rules, etc. Tripcom will most probably require a security architecture where application components process the security context and re-use middleware mechanisms to enforce policies defined in the application itself; e.g. Tripcom will require the formulation of security policies at run-time: for example, the

policies defining the access to a specific medical information might not just be specified in terms of categorization and predefined rules ("Lab record X is accessible to all medical personnel with role Y") but rather as part of the medical information itself. For example: the rule "If lab record X indicates drug use, such record may only be presented to role Y" requires applicative interpretation of the record.

This brings us to the discussion of Role based access control (RBAC) mentioned previously. RBAC proposed and developed by NIST[3] is around since 1993. All database systems and modern file systems implement RBAC concepts, although features like "static separation of duties", meaning the avoidance of conflicting role assignments through constraints or "dynamic separation of duties", meaning the avoidance of conflicting roles within a single user session through constraints applied at runtime, are seldom implemented. In the context of middleware for TripCom, the OASIS implementation of RBAC principles in the form of the XACML standard ("Extensible Access Control Markup langauge" [2] is of interest. XACML defines:

- an access control policy language: the policy language is used to express access control policies (who can do what when);
- a request/response language: the request/response language expresses queries about whether a particular access should be allowed (requests) and describes answers to those queries (responses);
- a conceptual architecture whose main new elements are a "Policy Enforcement Point" that enforces and executes the access policies, a "Policy Decision Point" that provides the underlying decisions at runtime and a "Policy Information Point" that holds the policies and access rules.

An open source, prototype implementation of XACML is offered by SUN Micrsosystems[4]. A middleware implementation of an XACML/RBAC system named "Hermes" with high relevance for TripCom is found in [15]. The authors describe a multi-domain, distributed publish/subscribe systems implemented on a web-of-trust. They describe the extension of their publish/subscribe middleware in order to process XACML based policies.

The CORBA security model allowed us to categorize the features of a middleware security architecture. J2EE compliant middleware (Application Servers) represent practical implementations of security architectures. It is therefore interesting to contrast these categorizations with existing J2EE security standards. Although Tripcom will most probably not build on top of J2EE compliant middleware — Tuplespaces are not part of the J2EE definition — the J2EE security standards give us an idea of the features that the middleware layer will have to implement.

3.6.3 Middleware security standards for J2EE

The following lists of JSR's (Java Specification Requests) impacts on security issues [60]:

- JSR 72: The Java GSS API.

[3]See `http://www.nist.org` for material related to RBAC.
[4]See `http://sunxacml.sourceforge.net` for details.

  - Aspect: confidentiality.
  - Content: provides a generic GSS-API for Java environments, allowing to connect various identification / authentication mechanisms into a Java environment.
  - Relevance for Tripcom: Medium, since Tripcom will most likely use standard mechanisms. However, the understanding of this JSR is important:
    * it provides the necessary base for the integration of pluggable authentication mechanisms;
    * GSS-API principles are crucial for the implementation of delegation mechanisms.
    * GSS-API is an IETF RFC (IETF GSS-API v1 (rfc1508/9-1993), IETF GSS-API v2 (rfc-2743/4-2000)), as well as an X/Open.

- JSR 105: XML Digital Signatures
  - Aspect: accountability.
  - Content: provides an interface in a Java environment for the creation and processing of digital signature in XML formatted content. Supports the W3C recommendation for XML-Signature and Processing (`http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/`).
  - Relevance for Tripcom: high; Tripcom will probably use signed content techniques. These interfaces should be implemented whenever signed content is processed, for example in Tuplespaces.

- JSR 115: Java Authorization Contract for Containers
  - Aspect: integrity.
  - Content: also known as JACC. Defines the implementation of policy driven authorization mechanisms for accessing J2EE based application containers. It is used to implement RBAC based policies. The specification is implemented in J2EE compliant application servers. It is the central security definition related to J2EE. The JSR defines three major architecture elements:
    * the policy configuration, e.g. a set of elements allowing to define authorization policies;
    * the policy provider decision point, e.g. the mechanism that responds to security related queries;
    * the policy enforcement point, e.g. the mechanism that enforces the security policies at the entry points to the application container.
  - Relevance for Tripcom: high; Tripcom will most likely use Role Based authorization schemes, e.g. the Tuplespace core will probably be required to implement — or to reuse — an equivalent mechanism.

- JSR 196: Authentication Service Provider Interface for Containers
  - Aspect: confidentiality.

- Content: allows to integrate standard or custom authentication service providers with J2EE application containers.
- Relevance for Tripcom: the interfaces specified in this JSR are "higher" than the GSS-API's from JSR 72. They are directly related to the application container. These interfaces might provide guidance for the implementation of authenticated access to components implementing a Tuplespace.

J2EE compliant components communicate via RMI or RMI/IIOP. A Security-Manager together with associated policy-objects protects the RMI mechanisms, in the sense that one component should be prevented from harming another component or remote system. The security concept behind RMI provides the base for trusted communication between components.

In summary, J2EE middleware implements not the full security complement of security mechanisms as modelled in CORBA. Basically, confidentiality and integrity mechanisms are provided by J2EE application containers. Accountability aspects are not merely represented — most J2EE Application server do not offer audit features.

3.6.4 Federation - the world of Triplespaces

The federation of a world of Tuplespaces over web-based protocols — or standard protocols in more general terms — is a major challenge for the security infrastructure. Out of the box middleware solutions provide only certain elements of the solution:

- identity management solutions,
- single sign-on solutions,
- provisioning solutions.

However, these solutions are not suitable for Tripcom:

- they maintain their own registries or stores in centralized locations (LDAP registries or other forms of user registries);
- they are essentially centrally managed or depend on centrally managed infrastructures (Domain systems);
- they use private, non-public protocols (nowadays replaced by SAML).

The vision of TripCom as a large public infrastructure mandates a federated architecture that supports multiple types of interacting user registries. It is therefore necessary to define a mechanism that allows any "service provider" (for example a specific tuplespace) to locate and interact with any "identity provider" within the public infrastructure. SAML ("Security Association Markup Language") [66] created by OASIS describes such mechanisms and provides a solid base for the creation of a federated architecture.

In the word of its authors "SAML, developed by the Security Services Technical Committee of OASIS, is an XML-based framework for communicating user authentication, entitlement, and attribute information. As its name suggests, SAML allows business entities to make assertions regarding the identity, attributes, and entitlements

of a subject (an entity that is often a human user) to other entities, such as a partner company or another enterprise application".

SAML covers all aspects required to define a federated system, based on the interaction of "service providers" with "identity providers":

- The protocol aspect: The protocol bindings specify the implementation on top of with SSL, SOAP and HTTP.
- The semantic aspect: The SAML core specifies the semantics of the assertion language with its entities like "Subjects", "Assertions", "Statements", "Conditions",... as well as the structure of assertion documents.
- The applicative aspect: the SAML profiles specify various standardized interaction patterns between "service providers" and "identity providers" in a federated system. For example:
  - Single Sign-On Profile for the implementation of a single sign-on from a web browser;
  - Enhanced Client Profile for the interaction with identity services according to the requirements of applicative services;
  - Request and security assertion profiles for the federation of security assertions;
  - Identity Provider Discovery Profiles and other Management profiles;
  - etc...

Although SAML is not a middleware in the proper meaning of the word, a middleware infrastructure should leverage on the SAML principles to support federation. A successful implementation of a scalable federated infrastructure leveraging SAML is the "Shibboleth" project created by the "Internet2 consortium" [3]. This project aims to provide a framework that uses SAML Single Sign-On profiles with an exchange of security attributes and assertions. The framework is used in various academic and educational projects.

3.6.5 Additional security standards related to middleware requirements

The basic security standards ISO/IEC 17799, ISO/IEC TR 13335 and ISO/IEC 15408 are as much concerned with compliance, audit and impact analysis as they are with the usual security aspects like authentication and authorization. A proper security architecture will therefore rely on middleware to provide capabilities like:

- audit the behavior of the system and demonstrate the compliance of the system to security policies,
- detect breaches of security policies,
- assess the impact of security breaches.

4 Trust and reputation

Reputation based trust management systems provide a mechanism by which an entity requesting a resource or willing to interact with some other entity may evaluate its trust in the reliability of the resource or in the transaction counterpart. This is achieved using reputation, given by opinions expressed by entities about other entities or resources. The word "trust" can be differently interpreted, as it depends on the context it is used in; we are dealing here with trust that a participant has towards somebody that "states something" and about statements. E.g., the target entity could be indifferently a Certification Authority that vouches for someone's identity, a participant who says that a restaurant is good in his opinion, etc. If a principal trusts the CA, this means that he considers reliable the statements the CA makes about the relationship among other users' public keys and their identities. If a principal trusts a friend as restaurant recommender, this means that he considers reliable his friend's statements about food. Trust is a matter dealing with a certain statement and the author of that statement, and reputation is a way that can be used in order to *establish and manage* trust. As an addition, some systems use social relationships between peers when computing trust and reputation values. In particular, they analyse a social network which represents the relationships existing within a community and draw conclusions about peer reputation based on different aspects of the social network.

Reputation information has been used with great success in at least two well known real cases. Both the eBay auction site [1] and the Google web search engine use reputation in order to provide their customers with better results. The former uses ratings left by eBay members after buying and selling goods. These comments and ratings are valuable indicators of a member's reputation as a buyer or seller on eBay. They are included, along with an overall feedback score, in the member profile and are centrally maintained by eBay. On the other side, Google uses the links that weave through pages in the WWW as a mean to rank pages [58], realising a reputation system dealing with resources.

As the TripCom project is addressed to a large scale distributed infrastructure with no authorities that can control it overall, centralized solutions such as those presented above are not well suited in it. An inspiration can come from reputation systems proposed for distributed environments, as peer-to-peer infrastructures are.

4.1 Adversaries and countermeasures in P2P reputation systems

In general, a reputation system assists participants in choosing a counterpart for a transaction or a resource. The two primary types of adversaries in P2P networks are selfish peers and malicious peers. We can assume that in TripCom the first kind of users will be a minor issue, as we will have clients and a set of distributed servers (differently from file sharing networks in which every participant is also a resource provider). The goal of malicious peers is to harm specific targets or the system as a whole. The following list briefly describes the general typologies of adversaries that could try to subvert the TripCom reputation mechanisms [47]:

Traitors are peers that may behave properly for a period of time, in order to build a good reputation, and then begin defecting. An example of traitors is given by

eBay merchants that participate in many small transactions, behaving correctly, and then use the good reputation they built in one or few fraudulent high-price transactions.

Colluders are peers acting together with a commonly agreed malicious goal. Behaving like this, they can cause more damage than each one acting independently.

Front peers are malicious colluding peers that cooperate with others in order to increase their reputation, and then provide misinformation in order to convince other peers of the good reputation of chosen malicious peers. This kind of attack is particularly difficult to prevent in an environment where there are no pre-existing trust relationships at bootstrap time.

Whitewashers are peers that purposefully create and use a new identity in order to shed any bad reputation they accumulated under their previous identity.

DoS attackers are peers that try to cause denial of service to the reputation mechanism.

In order to protect from attacks, the following phases must be carried out successfully by non-malicious peers [47]: the reputation system has to collect information on the behaviour of each peer (or about opinions expressed about resources), needs to score and rank other peers based on expected reliability, and finally, as the transaction has been completed, takes action against malicious peers while rewarding those that well behaved. The following considerations can be made about each phase.

Information gathering: associating a history of behaviour with a particular agent requires a sufficiently persistent identifier. Such an identifier should provide anonymity if required, should be spoof-resistant and unforgeable. Information integrity is another problem, and, as it is impossible to enforce honest and accurate reporting, many reputation systems simply do not attempt to directly verify the integrity of the collected information. Moreover, as the amount of reputation gathered increases, the credibility of each piece of information decreases. That's why the primary sources should be personal experience, external trusted sources, one-hop trusted peers, multi-hop trusted peers and finally global values. In [57] some strategies for choosing which data to collect are categorized (furtherly discussed in section 4.3).

Scoring and ranking: a peer's reputation may be computed on various input statistics collected from his history. What these statistics should be depends on various domain specific and general issues; however, the first thing to cope with is to define if only good behaviour can be tracked, or if also defection information is available. In this case, both good and bad behaviour should be used and the negative impact of bad behaviour should outweigh the positive impact of good behaviour. Anyway, should the peer's reputation be based only on the quality of the work it has done, or should it depend also on quantity? In [46] it is stated that a score that combines quality and quantity is more effective. Finally, the computed reputation value may be a binary value, a scaled integer or a value on a continuous scale. This choice is application dependent, although many "degrees" of reputation should be suggested.

Response: incentive schemes are needed in P2P systems in order to encourage co-operation. Usually they are most effective against selfish users than malicious users, and usually reward "good" users with prizes like bandwidth, quality of service etc. On the other side, punishment schemes (e.g. banning participants from the infrastructure) are more suitable against malicious peers.

4.2 Reputation systems

4.2.1 Trust-Recommendation Model

In [10] an approach based on combining a distributed trust model with a recommendation protocol is proposed. The focus of the approach is on the following four goals: decentralization, trust generalization, explicit trust, and recommendations. Decentralization allows each peer to take responsibility of its own trust policies and removes the need for those policies to be communicated to other peers. Thus it allows each peer to manage its own trust. Trust generalization is concerned with identifying that there are different dimensions to trust called trust categories, and trust in a peer varies depending on these dimensions. This is an important issue that needs to be considered in the TripCom project, as entities that can be trustworthy for some specific properties, maybe could be less (or not) trustworthy if other properties are considered. That's why in [10] trust needs to carry semantic meaning, so that values can be compared. Finally, in a large decentralized system, it may be impossible for a peer to have knowledge about all other peers. Therefore, in order to cope with uncertainty arising due to interaction with unknown peers, a peer relies on recommendations from known peers about these unknown peers. In this model, a trust relationship is always between exactly two entities, is non symmetrical, and is conditionally transitive. Mutual trust is represented as two distinct trust relationships. Two different types of trust relationships are distinguished. When one peer trusts another, it constitutes a direct trust relationship. But if a peer trusts another peer to give recommendations about another peers trustworthiness, then there is a recommender trust relationship between the two. Trust relationships exist only within each peer's own database and hence there is no global centralized map of trust relationships. Trust categories are used by peers to classify trust towards other peers depending upon which aspect of that entity is under consideration. A "reputation" is defined as a tuple consisting of a peer's name, the trust category and the specific trust value. A recommendation is defined as communicated trust information which contains reputation information.

4.2.2 P2PRep and XREP

The P2PRep protocol [26] is a widely cited proposal in peer-to-peer reputation literature. It provides a reputation-aware version of the Gnutella protocol by which a peer can assess the reliability of an offerer before actually downloading a resource from it. The mechanism is the following: after a peer downloads a resource from another one, it can record whether or not the download has been completed with satisfaction. Before downloading a resource, a peer can thus poll its peers about their knowledge of the offerer, thus assessing reputation. Being designed for an unstructured peer-to-peer network, the protocol itself is not much interesting for the TripCom project, but some design features are useful to report. First, as stated also in section 4.1, identifiers'

persistency must be assured. Second, care must be taken to ensure confidentiality and integrity of the messages used to exchange reputation data. To these purposes, the protocol uses public key encryption. In particular, a peer's identifier is chosen as the digest of a specific public key, obtained through a secure hash function (and for which the owner has the corresponding private key). Exchanged messages are then signed with a specific sender's secret key and encrypted with another specific recipient's public key.

An evolution of P2PRep, designed by nearly the same research group, is XRep. XRep combines peers' and resources' reputation, and thus some considerations reported in the XRep [28] paper may be useful to the TripCom project. A basic advantage of resource-based reputation systems is that votes actually express a property of the resource, therefore they can be seen as more reliable and can carry more semantics. On the other hand, resource-based solutions can only be applied when resources have an history; therefore, in scenarios like eBay, where most resources appear in the system only once, participants' instead of resources' reputation is essential. By considering both resources and participants reputation, advantage of both approaches can be found in the following fields[1]:

Reputations' life cycle: new resources offered by a "trustworthy" peer will be quickly regarded as reliable, avoiding cold-start problems. On the other hand, resource-based reputations may have a longer life cycle, as good resources can be recognizable as such regardless of who offers them.

Impact on peers anonymity: resource-based reputation can be used also in systems where peers are willing to take on a new fresh identity whenever they want.

Blacklisting: Peer-based reputation can effectively support blacklisting. With resource-based reputation, on the contrary, no reference is possible between a bad resource and the peer that provided it. Anyway, resource blacklisting can still be useful in stopping dissemination of malicious stuff.

4.2.3 EigenTrust

EigenTrust [42] is an algorithm designed for P2P networks that assigns each peer a unique global trust value, based on the peer's history of uploads. It is interesting to TripCom because, since it computes a single trust value, it could complement the subjective view that has been prescribed as a requisite, with a more general view. As stated in the EigenTrust paper, the network of peers can thus identify malicious peers and isolate them. Some important design considerations that led to the EigenTrust algorithm should be taken into account in the TripCom project:

- The system should be self-policing. That is, the shared ethics of the user population are defined and enforced by the participants themselves, and not by any central authorities (in TripCom, local authorities can do this only on the triple space they actually rule).
- The system should be able to maintain pseudonimity.

[1]Fields covered in [28] that are not relevant to the TripCom project are not discussed here.

- The system should not assign profit to newcomers.
- The system should have minimal overhead in terms of computation, infrastructure, storage and message complexity.
- The system should be robust to malicious collectives of peers.

In EigenTrust, the global reputation of each peer i is given by the local trust values assigned to peer i by other peers, weighted by the global reputations of the assigning peers. In a few words, we can say that EigenTrust defines an algorithm that aggregates local normalized trust values c_{ij} (resuming the number of satisfactory transactions that peer i had with peer j). Peer i asks opinions about peer j and weights them with the trust it places in those that express such opinions. This is a good way to have each peer gain a view of the network that is wider than its own experience. In order to get a wider view, the peer can ask his friends' friends, and so on. If it iterates this process, the trust vector representing the local trust values derived by peer i will converge to the same vector for every peer, thus representing a global trust vector (it will converge to the left principal eigenvector of the matrix $[c_{ij}]$). It should be emphasized that pre-trusted peers are essential to this algorithm, as they guarantee convergence and break up malicious collectives of peers. The distributed algorithm makes each node ask local trust vectors to some other nodes, and it is demonstrated that the algorithm converges quickly (for a network of 1000 peers, 10 iterations are sufficient to have values near the "true" global trust values). Moreover, in the TripCom project, we could imagine that values similar to those described above could be published in a Triple Space, reducing the problem of trust vector distribution (in EigenTrust, trust vectors are queried by means of messages, with consequent network load). Such vectors could be then protected as data are by the means of access rights or cryptography, as discussed in other sections of this document.

4.2.4 XenoTrust

XenoTrust [33] is the trust management architecture used in the XenoServer Open Platform [61], a public infrastructure for wide-area computing that could be an inspiration in facing TripCom open issues. [33] suggests that an event-based publish/subscribe methodology for the storage, retrieval and aggregation of reputation information can help exploiting asynchrony and simplicity. Some similarities can be found between TripCom and XenoServer, as the latter is an open and public infrastructure in which users can run programs and deploy large-scale experimental services. Users must be protected from intruders and snoops, as well as from other legitimate users that have malicious intentions. XenoTrust is structured as a global-scale federated system, whose constituent parts may have different notions of "correct" behaviour and where users are identified by a pseudonym. Each "XenoServer" hosts clients' tasks (some security-related analogy could be drawn among it and a Triple Space). A XenoServer is run by an authority (the XenoCorp) that is trusted and self-authenticating to its users. Authentication of other entities in the XenoServer world is carried out by credential issued to users by the XenoCorp they register with. Trust is managed by a two-level approach, by distinguishing authoritative and reputation-based trust. The former is the trust relationship users have with the XenoCorp they registered with, the latter is a discrete continuous property which quantifies the sub-

jective trustworthiness one component ascribes to another. Some interesting features of XenoTrust are:

- a statement is the basic unit of reputation information; it is represented as a tuple `{advertiser, subject, token, value(s), timestamp}` representing the identity of the advertiser and the subject of the statement, plus a token denoting the aspects of the subject reputation being considered, a series of values denoting the extent of reputation being considered and a timestamp. All of this is signed with advertiser's credentials;
- the computation of reputation vector is handled into the XenoTrust platform itself, allowing aggregation of information and incremental updating of values;
- participants can specify whether to use an event-based mechanism or a polling mechanism for reputation information retrieval;
- in order to accommodate a wide variety of different types of query, XenoTrust supports a simple atomic language for query rules definition. The rule-sets take the form `{principal, property, advertiser, function, [trigger]}`, where the first value is the subject of the query, "property" is the token that the query refers to, "advertiser" is a set of components whose advertisements are considered during query evaluation, "function" is the way reputation information is aggregated (e.g. min, max, mean, avg, etc.) and "trigger" is the threshold beyond which a user is notified about changes.

4.2.5 Regret

Regret [62] is an example of a reputation model considering both the social dimension of peers and their opinions. It is based upon three dimensions of reputation (individual, social and ontological), and combines them in order to obtain a single value of reputation. Individual reputation is formed in direct interactions with other peers. As each peer belongs to a group, social reputation relies on group relations. In particular, since a peer inherits the reputation of the group it belongs to, the group and relational information can be used to have some understanding of the behaviour of the peer when direct information is unavailable. Thus there are three sources of information that help a peer (say "A") in evaluating reputation of another peer (say "B"): the individual dimension between A and B, the information that A's group has about B (called "witness reputation") and the information that A's group has about B's group ("neighbourhood reputation"). Beyond this model, another feature that is interesting to the TripCom project is that Regret uses a "multi-faceted" reputation, in which the different types of reputation and the way they are combined to obtain new types of reputation are defined by the ontological dimension (e.g. the reputation of being a good flying company can combine the reputation of having good planes, of never losing luggage, of being timely, and all of these reputations maybe summarize the reputations of other dependent factors). Clearly, since reputation in subjective, each peer typically has a different ontological structure to combine reputations and could have different ways to weigh reputations when they are combined.

4.3 Interaction strategies

When reputation data are collected, or when the reputation system cannot provide any reputation data, how should agents behave in interacting with other agents? In [57], trust strategies (or tactics) that an agent can follow when interacting with others on the Semantic Web are discussed. These strategies represent the attitude an agent has in conditions of uncertainty. Five potential approaches are described in [57].

Optimistic systems accept others unless there is reason not to trust. If the benefits of cooperation are large and betrayal cost is low, the gain of such a strategy overcome the risks. Such systems are usually likely to benefit from decentralization. Moreover, some systems exploit optimism only for the purposes of a bootstrapping process.

Pessimistic systems are the opposite of optimistic ones, and pessimism corresponds to trust via personal acquaintance in the off-line world, which is the basic model of local trust. Anyway, in [57] it is stated that such a model of trust is not often capable of supporting very complex societies.

Centralised trust systems help users annotate information sources, like in eBay. Such systems can raise the question of how trustworthy should be these centralized trust warehouses. Moreover, usually this approach relies on some sort of implicit trust measurement mechanism. Anyway, centralization lays off the costs of interacting with and investigating agents to a central institution or authority, reducing the trust requirements for users.

Investigative approaches aim at reduce uncertainty by investigating or evaluating other agents to determine some salient details of operation. It is not a passive approach, as it actively tries to discover aspects of the environment that are relevant to reduce uncertainty, usually adopting statistic techniques (e.g. Bayesian networks).

Transitive approaches rely on the idea that an agent sends a message out about whether a potential agent is trustworthy. The network of acquaintances of that agent will then either send back an opinion based on experience, or pass the message onto its acquaintances, many of which will be unknown to the first agent. The aim is to increase the scope of an agents knowledge by exploring the community to bring in information from other, unknown, agents. However, trust is not strictly transitive, so small world theory and social network analysis techniques can be used to reduce the scope.

Summing up some considerations about the above strategies (expressed in [57]), optimism and pessimism meet most of the challenges half way, by avoiding most of the complexities of the situation. Bootstrapping is trivial with optimism, but very challenging with pessimism. Both approaches however, fail to take account of the context sensitivity of trust judgements. Investigation as a strategy can be very useful for bootstrapping trust, as some relevant properties of an agent may be known before it has actually begun to operate in a domain. Furthermore, investigation could meet the context challenge, assuming the methods of investigation were sensitive to the changes in context. However, as the infrastructure becomes more distributed the cost of investigation will increase largely. Centralisation is similar in some aspects

to investigation. If centralised authorities distribute certificates of trustworthiness in some domains that increase in complexity, it may be hard for such authorities to scale up. Users of competing authorities will need to decide between them, which may impose transaction costs. The most interesting systems are those that exploit transitivity, in that they allow relatively flexible responses. Up to now, where most transitivity systems break down most seriously is on context dependence.

5 Enterprise Application Integration use case

5.1 Introduction

As corporate dependence on technology has grown more complex and far reaching, the need for a method of integrating disparate applications into a unified set of business processes has emerged as a priority. After creating islands of automation through generations of technology, users and business managers are demanding that seamless bridges be built to join them. In effect, they are demanding that ways be found to bind these applications into a single, unified enterprise application. The development of Enterprise Application Integration (EAI), which allows many of the stovepipe applications that exist today to share both processes and data, allows us to finally answer this demand [29].

Enterprise Application Integration (EAI) is aimed at modernizing, consolidating and coordinating the applications in an enterprise. Typically, an enterprise has existing legacy applications and databases and wants to continue to use them while adding or migrating to a new set of applications that exploit the Internet, e-commerce, extranet, and other new technologies. Whereas portal technology represents a mainly presentational, user-facing component designed to aggregate the outputs from many applications and enterprise silos, EAI is chiefly concerned with connecting these applications

- Both legacy and green-field developments
- In a way that reduces development effort and preserves investment in legacy products

EAI allows diverse applications based on different technologies and platforms to be integrated into new and more tightly integrated super-applications. To this end, its primary functionality is to connect previously de-coupled applications, provide intelligent routing and transformation of messages in transit between these applications and give workflow capabilities so that automated and human decision-making processes can be integrated [29].

5.2 Requirements

Too often, security seems to be an afterthought to the implementation of new technology. In order to address security properly, it is necessary to build the system or, in this case, the integration solution from the ground up. It's important to remember that whenever *information is sent or received from enterprise systems, when interfaces to the systems are built, or when middleware is being implemented, security must be considered.*

Security must be considered early in the EAI project process. In most cases, EAI security will be *built on top of an already existing security structure* within the source and target applications (e.g., Top Secret, RACF, or Windows NT security) to be integrated. Therefore, in addition to integrating applications, EAI will need to integrate the security systems, because this paradigm doesn't take into account security [29].

There are a number of environmental considerations that impact the overall security architecture in an EAI environment. These are:

- The EAI infrastructure may not totally wrap the existing applications. In other words, there may still be other entry points into the application.
- The underlying applications in an EAI environment will almost certainly rely on different, standard or entirely proprietary, protocols and platforms.
- Each legacy application will almost certainly have considered security issues to some extent or another, thus integration becomes a problem of suitably aggregating these security models.

So the security system should work on different operating systems (Windows, Solaris), platforms (CORBA, J2EE), providing a common access management framework.

Security systems aims [67]:

Guarantee: information availability, integrity, confidentiality and auditability.

Avoid: information destruction, modification or revelation.

Decrease: information loss or risk of lost.

Security requirements typically require security-specific testing in addition to the traditional types of testing. Test cases may be based on misuse cases that are analogous to the test cases developed for use case based functional testing. Also, load and stress testing can be useful for testing Denial of Service (DoS) attacks.

Security in TSC involves system security and network security. System security refers to managing user access and authentication control, assignment of privilege, maintaining file and file system integrity, backups, monitoring processes, log-keeping, and auditing. Network security deals with protecting network and telecommunications equipment, protecting network servers and transmissions, combating eavesdropping, controlling access from untrusted networks, firewalls, and detecting intrusions [48].

We are going to group our requirements in eight topics :

1. **Authentication / Single-Sign On referring**

   Authentication is the process of determining whether someone or something is, in fact, who or what it is declared to be. Authentication is hot spot requirement on all EAI systems [44, 67]. The typical objectives of authentication requirement are to:

   - Ensure that externals are actually who or what they claim to be.
   - Avoid compromising security to an impostor

   Some requirements on authentication are:

   - A broad spectrum of authentication and gradual identification systems must be available, from simplest ones (limited security levels) until most advanced ones (high security levels).

- Define Entities and Users: In an EAI environment the concept of a user becomes more nebulous, thus entities such as business processes may need to have an identity for the purpose of defining their access permissions. However users with roles such as developer, administrator and manager need to be handled.
- User Single Sign-on: Users as well as applications will be interacting with each other in an EAI environment as well as interacting with the infrastructure itself (for the purpose of managing and monitoring). To this end, user experience, and thus SSO (Single Sign-on), is an important consideration.
- Server Single Sign-on: Session credentials and attributes are seamlessly passed between back end systems without the user being aware that this has occurred. In addition, the user identification should be unique. In other words, a super-user account should not be used for authentication to the back-end application.
- Cross Domain Single Sign On (CDSSO): This is essentially the same concept as single sign-on but encompasses the issue of applications being in different DNS domains. Typically this has been a problem in webtier environments where the protocols prevent state-management objects from being transferred between DNS domains. In a more general EAI sense CDSSO encompasses the problem of conveying appropriate security credentials between applications within loosely coupled-coupled organizations (i.e. business partners rather than units within a business). For all applications involved in one company this requirement is not necessary, but for communications enter-enterprises it is.
- Multiple authentication mechanisms: The authentication system should be able to support a wide variety of different mechanisms, for example NTLM (for internal users), LDAP, Databases, Radius and SSL client certificates. It is necessary for make the security platform extensible.
- Configurable authentication mechanisms: The authentication mechanisms should be highly configurable so that specific LDAP and directory schemas can be defined in order to perform user authentication.
- Simultaneous Authentication Mechanisms: Different sets of users have different authentication mechanisms, some will use LDAP to authenticate against; others will use SSL. Indeed during any given business process the session may be gathering an increasing set of credentials and dealing appropriately with delegation issues becomes important.
- Definition of authentication policies: It's possible to define authentication policies combining environment attributes. For example to define the following policies:
  - No users out side the local subnet can gain access during the weekend
  - Only users with the role of developer can gain access from a given local sub-net
  - Day of Week
  - Subnet IP address (e.g. access only permitted from internal networks on 10.2.*.*)

- Logouts and session control: Entities should be able to logout of their session and require re-authentication to gain access to the system again. The system should provide an inactivity timer, and if it expires to automatically log the user out.
- Unified User Repository: Either a single User Repository should be used, or automatic synchronization should occur. In any case two separate user management facilities must not be used.

Authentication requirement is also requested in a TS (Tuple Space) environment. Before accessing to TS the users must authenticate themselves (and the TS system MUST verify the authentication).

2. **Authorization referring** [44]

   It's not enough to just identify users and then give them access to everything. Instead, we need to have granular controls that give some users access to an object and prevent others from accessing the same objects. Proper granularity is important. When contemplating file-based security, the granularity is usually access rights such as read, write, execute, delete, create, etc.

   Some requirements on authorization are:

   - Role Based Access Control — users: Access to any and all resources should be controlled by a combination of user names and roles (as a minimum). Users may have a requirement to have more than one role, and to be in more than one at a given point in time.
   - Role Based Access Control — entities: Extends the concept of role-based access control, as applied to users, to be applicable to other entities in the EAI. This includes having individual application instances having an "identity" such that they may be restricted in what they do depending on whether they are acting on their own behalf or on behalf of a user. Examples include:
     - Is a particular application allowed to manage or monitor another application using EAI provided facilities?
     - Can an application route messages to another application?
     - Is an application allowed to see or browse for another application?
   - Dynamic Policy with environmental attributes: As for authentication, environmental attributes should be used in controlling whether an entity or user has access to a resource. In addition, the access decision could be made given a dynamic attribute of the entity/user, the attributing varying in real time or changing every session.

   The TS administrator will be in charge of set the policies that decide who can do what and in what conditions. For example these policies would determine who can create a new triple and who cannot. The access rights should be define based on the different roles of the scenario (the roles will depend on the final scenario chosen for EAI prototype), but perhaps in any circumstance perhaps is better to define them for single triples.

   The actors of a typical EAI scenario (although Tripcom scenario is been defining now, we assume it will have the same) are the computer users that act as clients

of the systems (use the capabilities of the system to do their business), and final users, that use the system as service client. The first will need different authorization than the second one, because they use the system for different purposes. Other important role in this scenario will be administrator, which will have the majority of the rights to use the system, although different granularity is good in order to avoid possible problems. Anonymous access has not sense if the system is for business, anonymous access has not profit, unless to get information

The TS MUST guarantee the access is according with each authorization profile. Also a TS policy MUST be defined, because some operations could not be performed by anyone or it is permitted an anonymous access. A mechanism of trust establishment in TS SHOULD be provided, giving the final users the credential data they need to access TS.

3. **Data Protection (Encapsulation)** [44]

   Data protection is a security requirement that ensures that the data haven't been tampered with en route. Data protection requires securing both data integrity and privacy. It's worth mentioning that data protection doesn't guarantee the identity of the message sender.

   Some requirements on data protection are:

   - Secure communication: by encrypting the entire message. Also the sender must trust all intermediaries. The secure communication provide data integrity (ensure data is not changed when it is transmitted)
   - Secure message: end to end message security independent of transport, supporting multiple protocols and multiple encryption technologies
   - Safe storage: Data couldn't be changed or couldn't be lost
   - The information that a customer supply has to be confidential. Mechanisms for ensuring that all imported and generated data are safe and confidential

   TSC MUST provide methods to guarantee integrity of messages, avoiding malicious altering: attackers altering bank accounts or forging identity.

4. **Nonrepudiation** [67]

   Nonrepudiation guarantees that the sender of a message is the same as the creator of the message. This is very useful for Web services used in the B2Bi domain, as otherwise a malicious sender can later disavow having created and sent a specific message

   The requirement is:

   - The message creator must be guaranteed by the system

   The users must not be able to repudiate what they have written in TS. So TS SHOULD ensure (perhaps by authenticating the users when they access to the Triple Space and perhaps with additional data) the activities of each user, and also who create the triples.

5. **Accounting** [44]

   A similar challenge exists when considering the requirement for accounting the means by which a Web service provider can store an audit trail of requests to the service(s). To satisfy this requirement, digital signatures are used. The signature ensures the integrity of the data. In addition, when used in conjunction with identity services, it conclusively associates the sender with the data that was sent.

   The requirement is:

   - The system must provide security alarms and the possibility of verify accounts.

   The TS SHOULD be able to log what users do, triple Space should be able to log any kind of data access.

6. **Immunity**

   Immunity guarantees that a system is able to protect itself from external attacks. The requirement is:

   - The system must protect itself.

   TSC MUST provide methods to protect from external attacks.

7. **Intrusion detection**

   Intrusion detection is the act of detecting actions that attempt to compromise the confidentiality, integrity or availability of a resource.

   The requirement is:

   - The system must have a mechanism of intrusion detection.

   TSC MUST provide methods to guarantee integrity of messages, avoiding malicious altering: attackers altering bank accounts or forging identity.

8. **Survivability**

   Survivability is the quantified ability of a system, subsystem, equipment, process, or procedure to continue to function during and after a natural or man-made disturbance; e.g. nuclear electromagnetic pulse from the detonation of a nuclear weapon. DoS (denial of service) attacks are a man-made disturbance. DoS attacks, which come in many forms, are explicit attempts to block legitimate users' system access by reducing system availability. [37]

   The requirement is:

   - The TS system MUST have a mechanism to ensure the correct functionality in case of disaster.
   - The TS system MUST have a mechamism to prevent DoS attacks.

5.3 State of the art

It is important to take into account that EAI framework hasn't any security standard. The EAI solutions only adopt some techniques or standards to provide some aspects of security [29]. The EAI solutions only adopt some techniques or standards to provide some aspects of security.

The security systems should be developed having in mind that the system should:

- Ensure that users and client applications are identified and that their identities are properly verified.
- Ensure that users and client applications can only access data and services for which they have been properly authorized.
- Detect attempted intrusions by unauthorized persons and client applications.
- Ensure that unauthorized malicious programs (e.g., viruses) do not infect the application or component.
- Ensure that communications and data are not intentionally corrupted.
- Ensure that parties to interactions with the application or component cannot later repudiate those interactions.
- Ensure that confidential communications and data are kept private.
- Enable security personnel to audit the status and usage of the security mechanisms.
- Ensure that applications and centers survive attack, possibly in degraded mode.

In the following lines, some techniques used for each requirement are going to be described, taking into account the specific objectives of each one of them.

1. **Identification, authentication and authorization**

   They are provided by the same techniques, because not only a person or system identification is not enough, but also it is needed to ensure the identity and to give certain privileges depending on the identity.

   So while an identification requirement identify the externals before interacting with the system, an authentication requirement verify externals identification assuring that they are who or what they claim to be and an authorization requirement specifies access and usage privileges of authenticated users and client applications.

   The typical objectives of an authorization requirement are to:

   - Ensure that one or more persons (who have been properly appointed on behalf of the organization that owns and controls the application or component) are able to authorize specific authenticated users and customer applications to access specific application or component capabilities or information.

- Ensure that specific authenticated externals can access specific application or component capabilities or information if and only if they have been explicitly authorized to do so by a properly appointed person(s).
- Thereby prevent unauthorized users from:
  - Obtaining access to inappropriate or confidential data.
  - Requesting the performance of inappropriate or restricted services.

The most common identification-authentication techniques are:

- **Userids and Passwords** : The most commonly used technique for authentication is userid/password pairs. This information is sent across the network in clear text or scrambled form; consequently, userid/password schemes are subject to replay attacks [32]
- **Challenge/Response and Shared Secrets** : A userid and password are presented to the target system and, if accepted, the system responds with a challenge to the originator. The originator then runs a special algorithm against the challenge and responds back to the target system. Another technique in this category is shared secrets or symmetric key technology.
- **Public Key Infrastructure (PKI)**: PKI technology is widely used on the Web with Secure Sockets Layer (SSL) communication. The basis for PKI is a certificate that An authorized Certificate Authority (CA) generate. When PKI technologies were first deployed, there were only a few CAs in existence, but today, we see companies using PKI technologies internally. They're becoming their own CA. SSL by itself is not enough, but PKI and SSL together are a perfect fit for Message Oriented Middleware authentication.

As said before, it's not enough just to identify users and then give them access to everything. Instead, it is needed have granular controls that give some users access to an object and prevent others from accessing the same objects. Proper granularity is important. When contemplating file-based security, the granularity is usually access rights such as read, write, execute, delete, create, and so on.

On the other side some authorization techniques (the person/services has been authenticated) can be [67]:

- Programatic (authorization lists or databases).
- Tokens-based (X509 tockens for example).
- Policy based (WSE: Web services enhancement).
- Role-based
- Kerberos: Kerberos is a network authentication protocol. It is designed to provide strong authentication for client/server applications by using secret-key cryptography

2. **Immunity**

   An immunity requirement is any security requirement that specifies the extent to which an application or component shall protect itself from infection by unauthorized undesirable programs (e.g., computer viruses, worms, and Trojan horses).

The typical objectives of an immunity requirement are to prevent any undesirable programs from destroying or damaging data and applications.

Security mechanisms for immunity are:

- Comercial antivirus programs.
- Firewalls.
- Prohibition of type-unsafe languages (e.g., C) that may allow buffer overflows that contain malicious scripts.
- Programming standards (e.g., for ensuring type safety and array bounds checking).

3. **Integrity** (part of data protection requirement)

   An integrity requirement is any security requirement that specifies the extent to which an application or component shall ensure that its data and communications are not intentionally corrupted via unauthorized creation, modification, or deletion.

   The typical objectives of an integrity requirement are to ensure that communications and data can be trusted.

   Security architecture mechanisms:

   - Cryptography/encryption algorithms.
   - Hash Codes.

   These algorithms include Data Encryption Standard (DES), Rivest-Shamir-Adleman (RSA), Pretty Good Privacy (PGP), Diffie-Hellman, etc. Checksums are extremely good at detecting single bit transformations, but are not as reliable for other kinds of modifications. Some products even provide for what's known as Forward Error Correction (FEC). Here, the product detects changes and recovers from the alteration by putting the bits back in the correct order. Another prevalent tamper-detection technique is a digital signature such as Message Digest 5 (MD5), a standards initiative of the World Wide Web Consortium (W3C).

   Even common e-mail using Secure Multimedia Internet Mail Extensions (S/MIME) can detect when a piece of mail was altered.

4. **Intrusion detection**

   An intrusion detection requirement is any security requirement that specifies the extent to which an application or component shall detect and record attempted access or modification by unauthorized individuals.

   The typical objectives of an intrusion detection requirement are to:

   - Detect unauthorized individuals and programs that are attempting to access the application or component.
   - Record information about the unauthorized access attempts.
   - Notify security personnel so that they can properly handle them.

   Security architecture mechanisms are:

- Alarms.
- Event Reporting.
- Use of a specific commercial-off-the-shelf (COTS):
  - Intrusion Detection System (IDS). Based on anomaly detect with threshold or Heuristic techniques.
  - Intrusion Prevention System (IPS).

5. **Nonrepudiation**

   A nonrepudiation requirement is any security requirement that specifies the extent to which a business, application, or component shall prevent a party to one of its interactions (e.g., message, transaction) from denying having participated in all or part of the interaction.

   The typical objectives of a nonrepudiation requirement are to:

   - Ensure that adequate tamper-proof records are kept to prevent parties to interactions from denying that they have taken place.
   - Minimize any potential future legal and liability problems that might result from someone disputing one of their interactions.

   Security mechanisms that can be used to implement this requirement are:

   - Digital signatures (to identify the parties).
   - Timestamps (to capture dates and times).
   - Encryption and decryption (to protect the information).

   There are XML security languages for Web Services, for example, SAML (Security Assertion Markup Language) or WS-Security from OASIS.

6. **Privacy** (part of data protection requirement)

   A privacy requirement is any security requirement that specifies the extent to which a business, application, component, or center shall keep its sensitive data and communications private from unauthorized individuals and programs.

   The typical objectives of a privacy requirement are:

   - Ensure that unauthorized individuals and programs do not gain access to sensitive data and communications.
   - Provide access to data and communications on a "need to know" basis.
   - Minimize potential bad press, loss of user confidence, and legal liabilities.

   Architectural security mechanisms are:

   - Public or private key encryption and decryption.
   - Commercial-off-the-shelf cryptography packages.

   In the case of B2Bi projects, the messages corresponding to Web service request and response should always be encrypted, using for example cryptography, secured socket layer (SSL), and so on.

7. **Accounting**

   A security auditing requirement is any security requirement that specifies the extent to which a business, application, component, or center shall enable security personnel to audit the status and use of its security mechanisms.

   The typical objectives of a security auditing requirement are to ensure that the application or component collects, analyzes, and reports information about the:

   - Status (e.g., enabled vs. disabled, updated versions) of its security mechanisms.
   - Use of its security mechanisms (e.g., access and modification by security personnel).

   Security mechanisms are:

   - Audit Trails.
   - Event Logs.

8. **Survivability**

   This requirement is any security requirement which an application or center shall survive the intentional loss or destruction of a component.

   The typical objective of a survivability requirement is to ensure that an application or center either fails gracefully or else continues to function (possibly in a degraded mode), even though certain components have been intentionally damaged or destroyed.

   Architectural security mechanisms that are:

   - Hardware redundancy.
   - Data center redundancy.
   - Failover software.

   Also in this requirement we have included DoS attacks. There are several DoS attack detecion techniques, among them: [11] [37]

   - ICMP Traceback (or iTrace)

     A router under the iTrace framework will pick one packet, statistically, out of a large amount of data packets (e.g.,20,000), and generate a special ICMP packet toward the same destination of the selected packet. When a victim is under a DDoS attack, a large quantity of bandwidth-consuming packets will be sent toward the victim anyway. Thus, "hopefully" the chance for routers along the attack paths picking the right packets to generate "iTrace" is big enough to trace the location of DDoS slaves

   - Packet Marking

     Instead of having routers send separate messages for the sampled packets, it is possible also to inscribe some path information into the header of the packets themselves.

- Activity Profiling

  Monitoring a network packet's header information offers an activity profile. Loosely defined, this activity profile is the average packet rate for a network flow, which consists of consecutive packets with similar packet fields (such as address, port, and protocol).

- Ingress Filtering

  Incoming packets to a network domain can be filtered by ingress routers. These filters verify the identity of packets entering into the domain, like an immigration security system at the airport

6 e-Health use case

The eHealth scenario reported in D8b.1 aim at making the TSC infrastructure suitable for supporting a European patient summary that can enable access and sharing of citizen clinical profile, anywhere and anytime among European Member States. This chapter reports the main security requirements that the TSC infrastructure should address in order to support such scenario.

A patient summary is a concise clinical document that aims to make patient's medical information available for unexpected contacts, or to comply with shared clinical pathways. It is made up of different sections that report both the ordinary patient registry data and the various medical aspects related to the clinical status of the citizen, such as: allergies, immunizations, psychological problems, medications, patient and relative medical history.

The interests on European Patient Summary are focused on providing a virtual common infrastructure for accessing critical citizens' health data with respect to interoperability, multi-lingualism and security-privacy compliance issues. Such a summary can be written by any authorized healthcare organization around Europe and can permit all EU citizens to feel free to move around Europe, assuring them that authorized healthcare organizations will be able to easily and securely access their health data stored in the summary.

The likelihood that unauthorized use will be made of information is a function of its value and the number of people who have access to it [13]. In this kind of scenario a very large number of users and data will be treated making privacy of citizen very critical. In particular, every citizen is the only owner of his personal clinical information and only authorized clinicians may have access to his record in order to avoid any unauthorized use of the data.

6.1 Actors, Objects, Operations

The *actors* of this scenario are the computer users that operate in a healthcare organization (such as general practitioners, specialists, emergency doctors, nurseries, clinical researcher, health administrators...) and computer programs that acts on behalf of one of this user (such as a batch program that perform a sort of statistical analysis that may impersonate the credential of a clinical researcher). It's unlike that a citizen can be a direct actor of this scenario. Nevertheless he is the owner of his health information and he delegates a clinician (usually his general practitioner) to be the responsible of the treatment of his clinical data.

The *objects* are the data held in every patient summary, which could be structured in several sections and subsections.

The *operations* that the actors can perform on the clinical data include the ability to read and write specific sections of the patient summary. No actors can delete or replace data; only incremental modifications are permitted. Anyway, the any citizen could ask his personal healthcare responsible to prevent read access onto specific sections of his summary from other clinicians. This measure aim to respect the privacy of each citizen according to his wishes. Moreover, any citizen could ask his responsible doctor to know the list of all the other users who and when has read or write any part of his patient summary, including any modification to the access rights.

6.2 Security Requirements

This section reports the main security requirements that the development of an European Patient Summary application demands to the Triple Space.

6.2.1 Citizen consensus

It's necessary that the citizen provides the official consent for the treatment of his personal data to his responsible doctor. With this consent, the citizen delegates the doctor to initialize the citizen's summary in the European Patient Summary. Patients must be made aware that information may be shared between other members of a care team (such as a general medical practice' or hospital department). Moreover, in emergency situations, the emergency doctors can access to the data of a patient (after identified it in the EPS) even if the patient had not explicitly authorized the doctor to do this. Certainly, this access will be recorded in the summary of the patient as an emergency access.

6.2.2 User authentication

Each user must be authenticated before accessing to any data in the system. Moreover, a user might want to have a number of different authentication credentials (e.g. where a doctor works in a hospital, a prison and a general practice); some of these will be signed by organisations, and others might not be (e.g. for her private practice). Users can be organized in roles and each user can be part of more that one role.

6.2.3 Access control

Each section of the patient summary must be marked with an access control list naming the people or groups of people who may read it and append data to it [13]. The system shall prevent anyone not on the access control list from accessing the record in any way. For example, the general practitioner in charge of a patient could have read and write access to any section of the summary whereas a radiologist should be only able to read a part of a patient history that regards the broken bones without having access to the patient's psychological problems.

6.2.4 Persistence

No-one shall have the ability to delete clinical information and any modification must not replace the previous state of information. Eventual clinical mistakes could be indicated by labelling the section with special warnings.

Anyway, if a citizen decides to withdraw his consent to the system, the general practitioner should be able to tagging the whole summary as no more accessible.

6.2.5 Responsible User

One of the clinicians on the access control list must be marked as being responsible. Only he may alter the access control list, and he may only add other health care professionals to it.

The citizen could be able to change his personal responsible and the data became inaccessible to the old responsible and accessible to the new one.

6.2.6 Notification

Any citizen must be informed of all the people who get access to his record [13]. This could be done by the responsible clinician that must notify the patient of the names on his record's access control list when it is opened, of all subsequent additions, and whenever the responsibility is transferred.

Moreover, there shall be effective measures to prevent the aggregation of personal health information. In particular, patients must receive special notification if any person whom it is proposed to add to their access control list already has access to personal health information on a large number of people.

6.2.7 Auditing

All accesses to clinical records shall be marked on the record with the user's name, as well as the date and time of the access event [13]. This has two major advantages: firstly, it's compliant with the capabilities to notify the citizen about each access to his clinical data and, secondly, this will presents any intruder with a credible chance of being caught.

6.2.8 ComSec

ComSec mechanisms ensure that access controls are not circumvented when sensitive data is sent from one computer to another. Moreover, the integrity and availability of medical information are also important, for the obvious safety and medico-legal reasons and ComSec requirements can protect the integrity of data sent through a network.

7 Requirements summary

In this chapter we summarize security requirements for TripCom, as emerged in the previous chapters.

7.1 General requirements

7.1.1 Triple space ownership and authority

- Each TS must have an owner, that is an authority that sets security rules and policies upon it.
- The TS owner may delegate administrative tasks to other entities, for the whole TS or for parts of it.
- The TS owner is an administrative authority, and does not state whether information published in the space is trustworthy or not.
- The world of triple spaces does not have a single owner.

7.1.2 Identities and authentication

- Agents (users, processes, services...) interacting with the TS must be able to authenticate themselves, using appropriate credentials.
- Unauthenticated access must be supported.
- Both direct and indirect authentication must be supported.
- Multiple hierarchies of certification authorities, as well as "web of trust" approaches like PGP, must be supported.
- Different TSs may "federate" themselves in order to use the same identity information and/or authentication process.
- The agent identity may be bound to a "real world" identity or may be just a pseudonym with no such link.
- Agents must be able to authenticate the TS infrastructure.

7.1.3 Access control

- An "information atom", i.e. the object of access rights, must be defined. This may be an entity used also for other purposes, or defined just for security purposes.
- Each atom must have an owner, i.e. the agent that has the right to set access permissions on it.
- Roles must be supported.
- An agent must be able to delegate another agent to act on its behalf.

- Access control policies, even if handled in a special way rather than like any other information, should be expressed using triples.

7.1.4 Other general requirements

- Besides relying on infrastructure security enforcement, agents must also be able to encrypt data before publishing.
- There must be a means for agents with the correct decryption keys to access the respectively encrypted data tuples.
- Agents must be able to interact using the TS without letting other agents, not involved in the communication, know their identities (communication anonymity towards third parties).

7.2 Infrastructure requirements

- The infrastructure must keep log and audit information about every access. Only auditors can access the audit. Local audits are maintained on every TS. Time-synchronization between audit and logs in the distributed system is sufficiently precise to allow the reconstruction of a chain of events throughout the whole system.
- Agents must not have direct access to the persistent storage layer of the Triple Space. This must be allowed only to authorized infrastructure components (middleware), or to special data source administrators (which are not TS agents, and operate outside the TS infrastructure).
- Some sort of security layer is necessary between the transport layer of the agent communication and the space itself.
- No agent access to the space should be possible outside of the defined access primitives (API) and through the security layer (no direct access to data objects in the space).
- The security layer must be in itself secure.
- The system needs to be able to authenticate the systems providing authentication services.
- Means are needed to protect tuple data from tampering, e.g. falsifying tuple identifiers, inserting “rogue” tuples.
- Means are needed to protect the tuplespace from malicious attack, e.g. placing malicious data into a tuple inserted into the space, flooding the space with access requests.
- In a physically distributed system, encryption may be applicable to communication between kernels.

- In a logically distributed system, individual authorities have only partial security and trust information and will need to be able to delegate some security/trust checks to other authorities.
- Means are needed to protect the space data from malicious storage operators, preventing tampering from within.

7.3 Trust and reputation functionalities requirements

- Users (information authors) must be able to express their opinions about objects (information) and other users (information authors).
- Agents must be able to retrieve users' opinions, if allowed, in order to build subjective trust evaluation.
- Agents must be able to use external parameters like explicit trust, or the existence of contracts, to affect their trust evaluations.
- The reputation mechanism must be general-purpose, i.e. applicable to different properties.

7.4 Use case requirements

7.4.1 EAI use case requirements

This section reports on the security requirements implied by the EAI scenario. For more information, please refer to deliverable 8A.1.

The security systems should be developed having in mind that the system should:

- Ensure that users and client applications are identified and that their identities are properly verified.
- Ensure that users and client applications can only access data and services for which they have been properly authorized.
- Detect attempted intrusions by unauthorized persons and client applications.
- Ensure that unauthorized malicious programs (e.g., viruses) do not infect the application or component.
- Ensure that communications and data are not intentionally corrupted.
- Ensure that parties to interactions with the application or component cannot later repudiate those interactions.
- Ensure that confidential communications and data are kept private.
- Enable security personnel to audit the status and usage of the security mechanisms.
- Ensure that applications and centers survive attack, possibly in degraded mode.

So the requirements are:

- Authentication requirement is also requested in a TS environment.
- Before accessing the TS, users must authenticate themselves (and the TS system must verify the authentication).
- The TS administrator will be in charge of setting the policies that decide who can do what and in what conditions.
- TS should define access rights based on the different roles of the scenario (the authorization definition).
- TS must support authorities hierarchy.
- The TS must enable to define new roles, and propagate them according to the hierarchy of the authorities.
- The TS must support the modification of access rights associated to the roles defined for an authority and the propagation of such modifications towards the inheritors of the authority.
- The TS must guarantee the access is according with each authorization profile.
- A mechanism of trust establishment in TS should be provided, giving the final users the credential data they need to access TS.
- TSC must provide methods to guarantee integrity of messages, avoiding malicious altering: attackers altering bank accounts or forging identity.
- TS should ensure (perhaps by authenticating the users when they access the Triple Space and perhaps with additional data) the activities of each user, and also who create the triples.
- TS should be able to log what users do, TS should be able to log any kind of data access.
- The TS must permit users with auditing rights to retrieve the chronological list of operations performed on one or more sections of a specific instance.
- TSC must provide methods to protect from external attacks.

7.4.2 e-Health use case requirements

This section reports on the security requirements implied by the e-Health scenario. For more information, please refer to deliverable 8b.1.

- The TS must support the authorities hierarchy at European scale.
- The TS must support a unique global user that is the administrator of the e-Health application and has rights for the definition of authorities, nominate their administrator and manage the mediation rules inside the TS for dealing with data, message format and communication protocol heterogeneity.

- The TS must enable each authority to define new roles.
- The TS must be capable to define users and to authenticate them.
- The TS must be capable of support the assignment of each user to the correspondent authority.
- The TS must support the propagation of roles according to the hierarchy of authorities.
- The TS must support the extension of roles associated to an authority and the propagation of any new role towards the inheritors of the authority.
- Each user in the TS must have one or more roles associated.
- Users cannot assume more than one role per moment in the TSC.
- The TS must support the definition of rights for *read-only*, *read/write*, *change of permissions* and *auditing*. The read-only permission implies that only read operation are permitted; the read/write permission enables the user to both read and write, but the only possible form of write is appending (thus: no deletion and no overwrite are allowed); change of permissions implies the user can modify the permissions of some other users and auditing implies that the user can query the list of operations performed on the data.
- The TS must support the definition of a set of access rights in function of a section of the schema and a role.
- The domain specific application that will be built upon the TS must support the definition of a set of access rights in function of a section of a specific instance of the schema, a specific user and a specific temporary situation related to the instance of the schema.
- The TS must support the modification of access rights associated to the roles defined for an authority and the propagation of such modifications towards the inheritors of the authority.
- Any user who access the TS must be authenticated.
- Any single operation (read, read/write, change of permission, audit) executed by a user on a TS entry must be tracked in the TS. This track must include the following information: the type of operation (read, write, change of permission, audit), the date and time of the operation, the name of the user who performed the operation, the name of the authority of the user and the information that has been read or changed within the context of this operation.
- The TS must permit a user who has the *change of permission* right on a specific instance of a summary to modify and revoke the rights of another user on a specific entry of such summary.
- The communication between the external components (administration tool and eHR) must be kept confidential. No one should read or alter the information exchanged.

- The TS must guarantee the integrity of the data written by users. Any illegal modification must be signalled.
- The TS must support the domain application in the control of the authorization flow on the Citizen summary that moves from the administrator of the original country towards the administrator of the destination country. There must not be any instant where no user has authorization to access the citizen data.
- Any information used by the TS for dealing with the heterogeneity of coding systems, message formats and communication protocols must not be modified by any user except the global administrator of the e-Health application.
- The TS must support users who get the *change of permissions* right to read the access rights of other users on any specific field or section of the summary.
- The TS must be permit users with auditing rights to retrieve the chronological list of operations performed on one or more sections of a specific instance of the summary. Any item of this list provides the following information: the type of operation (read, write, change of permission, audit), the date and time of the operation, the name of the user who performed the operation, the name of the authority of the user and the information that has been read or change within the context of this operation.
- Any read or write operation performed by any users must be tracked together with the state of the information read or written.
- The TS must block any unauthorized request made by users.
- The TS must track and log the blocked requests for information.
- The TS must guarantee the availability of the clinical data stored in the summaries.
- The TS must verify that any request made by a researcher is authorized by a privacy administrator.
- The TS must block requests for information that it it cannot verify are authorized.
- The TS must track and log blocked requests for information.

8 Conclusions

This document discussed security and trust requirements and state-of-the-art for TripCom. After a high-level discussion about the presence and role of "authorities" ruling the triple space, and about the impact of this on security requirements, the document analysed in detail state-of-the-art for security of TripCom infrastructure and for trust and reputation, and discussed security issues and requirements from EAI and e-health use cases. Moving forward, this work will be the basis for the design of TripCom security and trust model and architecture, and should be taken into account for the design of the overall architecture.

References

[1] Ebay home page. http://www.ebay.com/.

[2] extensible access control markup language 2 (xacml) version 2.0 3 oasis standard, 1 feb 2005. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.

[3] Shibboleth home page. http://shibboleth.internet2.edu/.

[4] Security in a Web Services World: A Proposed Architecture and Roadmap. http://www.ibm.com/developerworks/library/ws-secmap/, Apr 2002.

[5] Web Services Federation Language (WS–Federation). http://specs.xmlsoap.org/ws/2003/07/secext/WS-Federation.pdf, Jul 2003.

[6] Web Services Secure Conversation Language (WS-SecureConversation). http://specs.xmlsoap.org/ws/2005/02/sc/WS-SecureConversation.pdf, Feb 2005.

[7] Web Services Trust Language (WS–Trust). http://specs.xmlsoap.org/ws/2005/02/trust/WS-Trust.pdf, Feb 2005.

[8] Web Services Policy Framework (WS–Policy). http://specs.xmlsoap.org/ws/2004/09/policy/ws-policy.pdf, Mar 2006.

[9] Martín Abadi and Roger Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15, 1996.

[10] Alfarez Abdul-Rahman and Stephen Hailes. A distributed trust model. In *Proceedings of the 1997 workshop on New security paradigms*, pages 48 – 60, New York, NY, USA, 1998. ACM Press.

[11] Ahsan Habib, Mohamed M. Hefeeda, and Bharat K. Bhargava. Detecting Service Violations and DoS Attacks. http://www.isoc.org/isoc/conferences/ndss/03/proceedings/papers/12.pdf#search=%22DoS%20attacks%20detect%20techniques%22.

[12] Craig Sayers Alan. Computing the digest of an rdf graph.

[13] Ross J. Anderson. A Security Policy Model for Clinical Information Systems. In *Proceedings of the 15th IEEE Symposium on Security and Privacy*, pages 30–43. IEEE Computer Society Press, May 1996.

[14] Mark Bartel, John Boyer, Barb Fox, Brian LaMacchia, and Ed Simon. XML–Signature Syntax and Processing. W3C Recommendation, Feb 2002.

[15] Andras Belokosztolszki, David M. Eyers, Peter R. Pietzuch, Jean Bacon, and Ken Moody. Rolebased access control for publish/subscribe middleware architectures. In *Proceedings of Second International Workshop on Distributed Event-Based Systems (DEBS'03)*, 2003.

[16] Christian Bizer and Radoslaw Oldakowski. Using context- and content-based trust policies on the semantic web. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 228–229, New York, NY, USA, 2004. ACM Press.

[17] B. Blakley. *CORBA Security: An Introduction to Safe Computing with Objects.* Addison Wesley, 2002.

[18] P.A. Bonatti, C. Duma, N. Fuchs, W. Nejdl, D. Olmedilla, J. Peer, and N. Shahmehri. Semantic web policies - a discussion of requirements and research issues.

[19] Brian Matthews and Theo Dimitrakos . SWAD-Europe: Deliverable 11.1: A Framework for Deploying Trust Policies on the Semantic Web, 2004. `http://www.w3.org/2001/sw/Europe/reports/trust/11.1/trustpolicies.htm#[9]`.

[20] C. Bryce and M. Creminini. Coordination and security on the internet. In *Coordination of Internet agents: models, technologies, and applications*, pages 274–298, London, UK, 2001. Springer-Verlag.

[21] C. Bryce, M. Oriol, and J. Vitek. A Coordination Model for Agents Based on Secure Spaces. In P. Ciancarini and A. Wolf, editors, *Proc. 3rd Int. Conf. on Coordination Models and Languages*, volume 1594, pages 4–20, Amsterdam, Netherland, 1999. Springer-Verlag, Berlin.

[22] Jon Callas, Lutz Donnerhacke, Hal Finney, and Rodney Thayer. OpenPGP Message Format. IETF RFC 2440 (Standards Track), Nov 1998.

[23] Jeremy J. Carroll, Christian Bizer, Pat Hayes, and Patrick Stickler. Named graphs, provenance and trust. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 613–622, New York, NY, USA, 2005. ACM Press.

[24] J.J. Carroll. Signing RDF Graphs. In G. Goos, J. Hartmanis, and J. van Leeuwen, editors, *Proc. of the 2nd Int'l Semantic Web Conf.* Springer Verlag, September 2003.

[25] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[26] Fabrizio Cornelli, Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi, and Pierangela Samarati. Choosing reputable servents in a p2p network. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 376–386, New York, NY, USA, 2002. ACM Press.

[27] M. Cremonini. *Security and Mobility Issues in Software Agent Systems.* PhD thesis, DEIS, University of Bologna, 2000.

[28] Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi, Pierangela Samarati, and Fabio Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 207–216, New York, NY, USA, 2002. ACM Press.

[29] David S. Linthicum. *Enterprise Application Integration*. Addison Wesley, 1 edition, 1999. ISBN 0-201-61583-5.

[30] S. Demurjian, K. Bessette, T. Doan, and C. Phillips. Concepts and capabilities of middleware security. In *Middleware for Communications, Qusay H. Mahmoud (editor)*. John Wiley & Sons, 2004, 2004.

[31] Tim Dierks and Eric Rescorla (Ed.). The Transport Layer Security (TLS) Protocol Version 1.1. IETF RFC 4346 (Standards Track), Apr 2006.

[32] Donald G. Firesmith. Engineering Security Requirements, 2006. Firesmith Consulting.

[33] Boris Dragovic, Evangelos Kotsovinos, Steven Hand, and Peter R. Pietzuch. Xenotrust: Event-based distributed trust management. In *DEXA '03: Proceedings of the 14th International Workshop on Database and Expert Systems Applications*, page 410, Washington, DC, USA, 2003. IEEE Computer Society.

[34] Blake Ramsdell (Ed.). Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification. IETF RFC 3851 (Standards Track), Jul 2004.

[35] Carl Ellison, B. Frantz, B. Lapson, R. Rivest, B. Thomas, and T. Ylonen. Spki certificate theory. Technical report, September 1999.

[36] David Ferraiolo and Richard Kuhn. Role-Based Access Control. In *15th NIST-NCSC National Computer Security Conference*, pages 554–563, 1992.

[37] Glenn Carl and George Kesidis. Denial-of-Service Attack-Detection Techniques, Jan/Feb, 2006.

[38] Object Management Group. Security service specification. http://www.omg.org/docs/formal/02-03-11.pdf, March 2002.

[39] Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, and Henrik Frystyk Nielsen (Ed.). SOAP Version 1.2 Part 1: Messaging Framework. W3C Recommendation, Jun 2003.

[40] Takeshi Imamura, Blair Dillaway, and Ed Simon. XML Encryption Syntax and Processing. W3C Recommendation, Dec 2002.

[41] L. Kagal, J. Undercoffer, F. Perich, A. Joshi, and T. Finin. A security architecture based on trust management for pervasive computing systems, 2002.

[42] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 640–651, New York, NY, USA, 2003. ACM Press.

[43] Stephen Kent and Karen Seo. Security Architecture for the Internet Protocol. IETF RFC 4301 (Standards Track), Dec 2005.

[44] Kurt Lingel. Security Requirements for Message-Oriented Middleware, 2001. `http://whitepapers.zdnet.com/SECURITY/`.

[45] Butler Lampson, Martín Abadi, Michael Burrows, and Edward Wobber. Authentication in distributed systems: Theory and practice. *ACM Transactions on Computer Systems*, 10(4):265–310, 1992.

[46] Sergio Marti and Hector Garcia-Molina. Modeling reputation and incentives in online trade. Technical report, Stanford University, 2004.

[47] Sergio Marti and Hector Garcia-Molina. Taxonomy of trust: Categorizing p2p reputation systems. *Computer Networks*, 50(4):472–484, 2006.

[48] Martin Murth. Security and Privacy Models in Triple Space, March 15, 2006.

[49] R. Matthew, R. Agrawal, and P. Domingos. Trust management for the semantic web, 2003.

[50] N. Minsky and V. Ungureanu. Unified support for heterogeneous security policies in distributed systems. *Proceedings of the 7th security conference. (USENIX Association: Berkeley, CA)*.

[51] Naftaly H. Minsky and Jerrold Leichter. Law-governed linda as a coordination model. In *ECOOP '94: Selected papers from the ECOOP'94 Workshop on Models and Languages for Coordination of Parallelism and Distribution, Object-Based Models and Languages for Concurrent Systems*, pages 125–146, London, UK, 1995. Springer-Verlag.

[52] Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, and Ronald Monzillo (Ed.). Web Services Security: SOAP Message Security 1.0 (WS–Security 2004). OASIS Standard 200401, Mar 2004.

[53] Dalit Naor, Moni Naor, and Jeffrey B. Lotspiech. Revocation and tracing schemes for stateless receivers. In *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pages 41–62, London, UK, 2001. Springer-Verlag.

[54] W. Nejdl, D. Olmedilla, and M. Winslett. Peertrust: automated trust negotiation for peers on the semantic web, 2003.

[55] B. Clifford Neuman and Theodore Ts'o. Kerberos: An Authentication Service for Computer Networks. *IEEE Communications*, 32(9):33–38, 1994.

[56] Rocco De Nicola, Gian Luigi Ferrari, and Rosario Pugliese. Coordinating mobile agents via blackboards and access rights. In *COORDINATION '97: Proceedings of the Second International Conference on Coordination Languages and Models*, pages 220–237, London, UK, 1997. Springer-Verlag.

[57] Kieron O'Hara, Harith Alani, Yannis Kalfoglou, , and Nigel Shadbolt. Trust strategies for the semantic web. In *Proc. of the Workshop on Trust, Security, and Reputation on the Semantic Web*, November 2004.

[58] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[59] Adrian Perrig, Ran Canetti, Dawn Song, J. D. Tygar, and Bob Briscoe. Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction. IETF RFC 4082 (Informational), Jun 2005.

[60] The Java Community Process. Java specification requests (jsrs). http://www.jcp.org.

[61] Dickon Reed, Ian Pratt, Paul Menage, Stephen Early, and Neil Stratford. Xenoservers: Accountable execution of untrusted programs. In *HOTOS '99: Proceedings of the The Seventh Workshop on Hot Topics in Operating Systems*, page 136, Washington, DC, USA, 1999. IEEE Computer Society.

[62] Jordi Sabater and Carles Sierra. Regret: reputation in gregarious societies. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 194–195, New York, NY, USA, 2001. ACM Press.

[63] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C, Second Edition*. Wiley, October 1995.

[64] Jennifer G. Steiner, B. Clifford Neuman, and Jeffrey I. Schiller. Kerberos: An authentication service for open network systems. In *Proceedings of the USENIX Winter 1988 Technical Conference*, pages 191–202, Berkeley, CA, 1988. USENIX Association.

[65] Andrew S. Tanenbaum, Robbert van Renesse, Hans van Staveren, Gregory J. Sharp, Sape J. Mullender, Jack Jansen, and Guido van Rossum. Experiences with the Amoeba distributed operating system. *Communications of the ACM*, 33(12):46–63, 1990.

[66] OASIS Security Service TC. Security assertion markup language (saml) v2.0. http://www.oasis-open.org/specs/index.php, March 2005.

[67] Telefonica I+D. Tecnologias de Seguridad, 2006. Internal documentation.

[68] Giovanni Tummarello, Christian Morbidoni, Paolo Puliti, and Francesco Piazza. Signing individual fragments of an rdf graph. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 1020–1021, New York, NY, USA, 2005. ACM Press.

[69] A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni, and J. Lott. Kaos policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement. In *POLICY '03: Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks*, page 93, Washington, DC, USA, 2003. IEEE Computer Society.

[70] Debby M. Wallner, Eric J. Harder, and Ryan C. Agee. Key Management for Multicast: Issues and Architectures. IETF RFC 2627 (Informational), Jun 1999.

[71] Larry Zhu, Paul Leach, Karthik Jaganathan, and Wyllys Ingersoll. The Simple and Protected Generic Security Service Application Program Interface (GSS-API) Negotiation Mechanism. IETF RFC 4178 (Standards Track), Oct 2005.