13 CONSIDERATIONS ON SECURE FIPA COMPLIANT AGENT ARCHITECTURE

Tomáš Vlček[1], Jan Zach[2]
[1] *Czech Technical University in Prague, Czech Republic; vlcek@labe.felk.cvut.cz*
[2] *CertiCon a.s., CAK, Prague, Czech Republic; zach@certicon.cz*

This paper is intended to support security instruments for FIPA (http://www.fipa.org) compliant architectures. These security instruments, should increase trust and confidentiality within and among agent communities/technology and provide security mechanisms, as encryption, authentication, message integrity services, etc., employing cryptography algorithms.

1. INTRODUCTION

In the paper proposed secure architecture is FIPA compliant in the sense that every common FIPA compliant agent should be able to interact with agents having this extended architecture even not knowing the security mechanisms. On the other hand, agents demanding for security can, of course, refuse plain, non secured communication with non authenticated agents, etc.

The security mechanisms can utilize existing agent platform mediators such as the Agent Management System (AMS), Directory Facilitator (DF), or Agent Communication Channel (ACC), e.g. for authentication purposes[1]. But these new tasks enforce to make requirements for these mediators much more thoroughness, as FIPA specifies them too loosely[2] for these purposes (FIPA 1998, FIPA 2000). Other approaches could be found in (Wulf 1995, Wong 1998).

This paper is aimed mainly to "static", i.e. non-mobile, agents, but it should be possible to extend the architecture by mobility support easily[3].

2. ARCHITECTURE DESCRIPTION

All the requirements defined as a data items for architecture design are grouped into General Requirements – see Table 1, and Security Requirements – see Table 2, and are labeled by Requirement Identifier (RID) for future reference.

The specification uses common abbreviations as well as the terminology of FIPA standards (FIPA 1998, FIPA 2000); definitions of basic terms used, e.g. Agent Platform (AP), Home Agent Platform (HAP), directory Facilitator (DF) etc. can be found in (FIPA, 1998).

V. Mařík et al. (eds.), *Knowledge and Technology Integration in Production and Services*

Table 1 summarizes explicit general requirements imposed on the architecture design from the point of view of its compliancy with the FIPA system standards, transparency, legacy systems compatibility, reliability and optimal performance.

Table 1 – Requirement Specification – General Requirements

ID	Requirement	
G1	The resulting architecture should fit into the FIPA compliant system standard [1]	
	G1.1	Security services should be transparent whether within or outside an Agent Platform (AP).
	G1.2	An agent residing on an insecure platform (i.e. with enabled secure mechanisms) should be able to receive insecure messages, e.g., from outside of the AP. Whether the agent is willing to respond in an insecure way or at all depends on the agent itself.
	G1.3	Messages should consist of a FIPA compliant, non encrypted envelope[4] extended by encryption information as well as of a plain or encrypted content of the message. Furthermore, authenticity of the envelope should be easily verifiable as well.
G2	The resulting architecture should be vendor, patent, proprietary algorithm independent.	
G3	The resulting architecture should be available and reliable.	
G4	All the necessary security infrastructure should be accessible within an AP. If any security service requires an inter-platform communication, this should be transparently provided by trusted AP's mediators.	
G5	For performance and maintenance reasons, the resulting architecture should enable security based on trusted relationships and thus request/service delegation. E.g., an agent authenticated in his Home Agent Platform (HAP) is trusted in an adjacent AP which shares the trusted relationship with this HAP.	
G6	All the security services should be transparent to an agent.	
G7	The security architecture should be easily applicable to and integrable with any legacy system.	
G8	The architecture should support administrative domains in order to avoid administration troubles.	
	G8.1	It should be possible to settle administrative domains in a hierarchical way so that one can be immersed to another.
	G8.2	In any domain it should be possible to specify groups and roles assignable to agents belonging to that domain.

Table 2 defines the basic set of security services related to the platform and agent security, security threads resistance, security polices and authentication mechanisms. Dependencies among particular architectural security services are depicted in Fig. 1.

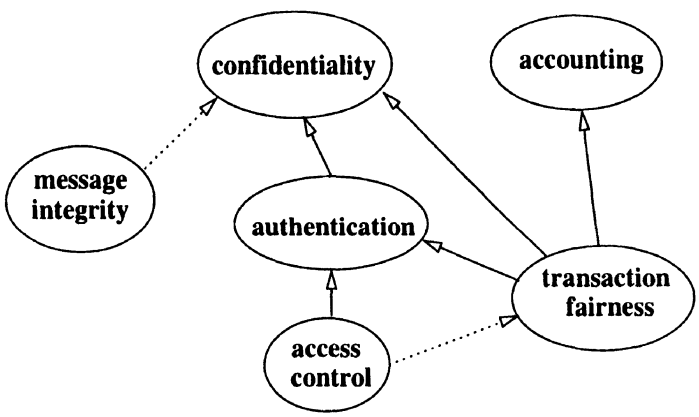

Figure 1 – Service Dependencies

Table 2 – Requirement Specification – Security Requirements

ID	Requirement		
S1	The architecture should support at least the following security services		
	S1.1	**Authentication Service**, i.e., an arbitrary agent should be able to prove its identity.	
	S1.2	**Confidentiality Service**. This service should protect communication against passive attacks as eavesdropping or traffic analysis.	
	S1.3	**Access Control Service** should support access control to agents' particular resources and alleviate agents' burden associated with it.	
		S1.3.1	The architecture should support access control to resources based on **roles** and **groups** assigned to a particular agent.
		S1.3.2	Roles and groups should be unique within a domain and all its subdomains.
		S1.3.3	An agent could have assigned an arbitrary number of roles and groups.
	S1.4	**Transaction Fairness Service** should provide protection of a particular transaction against repudiation or an attempt not to fulfill obligations, etc. In addition, it should serve as a trusted third party, i.e. as an arbitrator or transaction mediator, and support accountability and fairness within an agent community.	
	S1.5	**Accounting Service** should support logs and their analysis, monitoring, intrusion detection, etc. Besides it is required by the Transaction Fairness Service, it should improve availability and reliability of the system as a whole.	
	S1.6	**Message Integrity and Protection Service** should provide message protection against modification, alternation, errors, etc. This protection includes not only the content of a message but even its plain envelope. Easy detection of the message integrity should be possible.	
S2	The resulting architecture should be as much as possible resistant at least against the following security threads:		
	S2.1	Eavesdropping.	
	S2.2	Masquerading.	
	S2.3	Reply attack.	
	S2.4	Message alternation.	
	S2.5	Denial of service (DoS) and interruption.	
S3	An AP should be able to set up its security policy dependent on which **environment** it operates:		
	S3.1	**Secure.** The security is switched off.	
	S3.2	**Unsecure.** All security mechanisms are switched on.	
	S3.3	**Security Debugging.** Extensive logging is switched on.	
S4	An agent should be able to use at least the following security services:		
	S4.1	**Authentication Service.** All counterparts of communication should be able to prove their identity. If an unauthenticated message is received, the agent can throw it away or respond with an authentification request/failure.[5]	
	S4.2	**Access Control Service.** The service demander should prove its identity and rights to the service.	
	S4.3	**Message Integrity and Protection Service.** Messages should be protected against modification or other corruption during their transport. This option does not imply that the messages are encrypted.	
	S4.4	**Confidentiality Service.** All messages should be encrypted to prohibit their unauthorized access.	
	S4.5	**Transaction Fairness Service.** This service can be employed for assuring fairness during particular transactions.	
	S4.6	**Filtering.** This policy enables to filter incoming messages according to the source address, authentication, etc. Filtering should be performed within agents.	
S5	Resulting security level should be derived from security requirements of an agent and its HAP.		
S6	An AP should be able to force its agents to start using a security mechanism.		

S7	The architecture should be open for additional security services as availability, accounting, accountability, etc. These services are too sophisticated to be included into the basic security model but it should be possible to add them easily.
S8	There should be specified interfaces supporting both a symmetric, i.e. a private key, and an asymmetric, i.e. a public-key, encryption.
S9	An agent, accessing a service, should prove its identity, if its HAP is insecure, the service is not public, and/or the authentication is just required.
S10	Every agent should be able to authenticate at least in its HAP.
S11	Every service provider, including the public one, i.e. a mediator, running on an insecure AP and providing data should authenticate to its client.
S12	Identity of an agent should be proved by a time limited certificate issued by an authorization authority as the AMS or the PKI maintainer.

3. DESIGN DECISIONS

It seems reasonable to suppose an agent platform to be either secure or insecure. The decision about platform security (and security policy) is up to the platform owner. If an agent within a platform requires some security mechanisms (authentication, confidentiality, message protection, etc.) and its home platform does not support it, an error message is returned.

The inter- and intra- platform security should be distinguished because of considerable inherent differences: The inter-platform security concerns larger agent communities, e.g., the Internet, whereas the intra-platform security will be used in smaller communities, e.g., on a factory floor. It is apparent there are a bit different requirements in both cases concerning performance[6] and features they provide, e.g., agents on a factory floor can be supposed to provide much more prolific repertoire of services than their open-community counterparts, they could require the control access support for their particular services (not for the agent as a whole), etc.

There will be both the symmetric (shared secret key) and the asymmetric (public key) crypto algorithms employed depending on wether we deal with the inter-platform or the intra-platform security.

Inter-platform security will be provided by two means: trusted platforms and the public-key cryptography.

The term trusted platforms means that AMS's of both AP's share a secret key and they mutually trust in provided authentication information, i.e. certificates. This technique is available, e.g., within an enterprise, it is simple and does not require the sophisticated PKI infrastructure. Some potential issues can stem from the trust intransitivity. In addition, every platform or a trusted group of platforms can have its own public key which authenticates it within the agent universe.

In an open environment, e.g., the Internet, the public-key cryptography is much more applicable. The main advantage is the possibility of hierarchical certification, the main drawback[7] is the complexity of maintaining (and assuring security of) such hierarchical certificates. Of course, it is possible for a single agent to have its own public key, if necessary.

Intra-platform security requires the same security mechanisms as in the case of the inter-platform security but, in addition, some mechanism, e.g., the single service access control, should be provided. This can be done in a Kerberos-like manner. All security mechanisms should be implemented using the symmetric key encryption.

3.1 Trust Relation

To reason about trust within and among AP's we introduce the trust relation by the following definition:

Definition: As the **trust** it is meant a binary relation defined as

$Agent \times Agent \rightarrow \{true, false\}$

expressing that the first agent trusts or mistrusts the second agent. The symmetric trust will be denoted as $\Leftrightarrow_T$, the asymmetric one as $\Rightarrow_T$. The trust $Agent_1 \Rightarrow_T Agent_2$ can be established after the $Agent_2$ proves its identity, i.e. authenticates, to the $Agent_1$. This relation is not commutative.

In principle, the authentication can proceed in two manners as outlined in the following schema:

(i) The agent A and agent B share a secret key $K_{A,B}$, the authentication protocol then can look as follows:

1. $A \rightarrow B : E_{K_{A,B}}[nonce_A]$
2. $B \rightarrow A : E_{K_{A,B}}[nonce_A + 1, nonce_B]$
3. $A \Rightarrow_T B$

The protocol can further continue in the following way:

4. $A \rightarrow B : E_{K_{A,B}}[nonce_B + 1]$
5. $B \Rightarrow_T A, A \Leftrightarrow_T B$

(ii) The agent A holds its own, by a 3rd authority certified, public key:

1. $A \rightarrow B : nonce_A, Sgn_A[data\_verifiable\_by\_B]$
2. $B \Rightarrow_T A$

This protocol can continue:

3. $B \rightarrow A : Sgn_A[nonce_A]$
4. $A \Rightarrow_T B, A \Leftrightarrow_T B$

By a *nonce*, in the context of the protocols, it is denoted an arbitrary piece of data, the only condition that has to hold is that both the agents know the transformation *nonce+1* so that the agent which generated that nonce knows which value to expect after the transformation. The main reason for introducing a nonce is the countermeasure against the reply attack and a proof that the counterpart understands the message.

3.2 Authentication

Identity of an agent will be proved for other agents by a certificate issued by a **trusted authority** (TA). If the identified agent does not possess its public key, the certification authority will be its home TA, otherwise the certificate can be issued by any AP. The issued certificate will be valid within the authorizing TA's AP only. Part of the certificate will be a secret session key used for the consecutive session.

The following prepositions are assumed to be valid:

1. A TA trusts an agent if the agent proves its identity by a secret key (sometimes called the master key) shared between the agent and the TA.

The TA then attests this trust by issuing a **certificate**. After the certificate delivery, the proposition $agent\ @\ HAP \Leftrightarrow_T TA\ @\ HAP$ is valid. This type of certification will be possible only within the agent's HAP.

2. A TA trusts an agent if the agent proves its identity by its public key which is certified by the PKI authority. The TA then assets this trust by issuing a **certificate** and it holds $agent\ @\ HAP \Leftrightarrow_T TA\ @\ HAP$. This type of authorization is possible even across APs.
3. A certificate issued by a TA is issued for one specific agent. If the agent being authenticated needs to be authenticated to another agent, it needs a new certificate issued for that another agent.
4. An agent *a@HAP* trusts another agent *b@HAP* if that agent possesses a valid certificate issued by its home TA agent *ams@HAP*.
5. The trusted AP relationship means that the trusted APs share secret keys which allows their mutual authentication.
6. The shared secret key for AP's trust will be held by TAs of the trusted APs.
7. Trust is not transitive in general, transitivity, i.e.
   $(TA\ @\ AP_1 \Rightarrow_T a\ @\ AP_1) \wedge (TA\ @\ AP_2 \Rightarrow_T TA\ @\ AP_1) \wedge (b\ @\ AP_2 \Rightarrow_T TA\ @\ AP_2) \rightarrow$
   $\rightarrow (b\ @\ AP_2 \Rightarrow_T a\ @\ AP_1)$
   holds within the trusted platforms only, i.e. where $(TA\ @\ AP_1 \Rightarrow_T TA\ @\ A_2)$

In principle, we can distinguish the following authentication cases:

- **Authentication within an AP without a Public Key.** This is the most common generic way to authenticate within an AP, see Fig. 2(1). The TA issues a certificate to the agent *A*, authenticating *A* to *B*, as TA proved its identity by acquaintance of the shared secret key. The certificate contains a session key which identifies the agent *A* to *B*.
- **Authentication within an AP with a Public Key.** See Fig. 2(2). This method works as in the case 1, but the public key is used for authentication. Communication 2 and 3 serves for acquiring the valid public key to verify the agent's identity.
- **Authentication in a trusted AP without a Public Key.** See Fig. 2(3a). In this case, the scenario is: 1. *A* acquires an authentication certificate issued by the home AMS and 2. In virtue of the certificate, *A* acquires another certificate valid in the adjacent trusted platform AP_2.
- **Authentication in a trusted AP with a Public Key.** See Fig. 2(2), or 2(3b).
- **Authentication in a non-trusted AP with a Public Key.** See Fig. 2(2). This is the most general way to get the certificate.

In all the mentioned cases, the result is a valid authentication certificate usable in the destination platform. A direct authentication with a public key, i.e., without any mediation with an AMS it is not prohibited but it is not recommended.

3.3 Access control

The access control (i.e. authorization) to agent's resources will be provided by the following means:

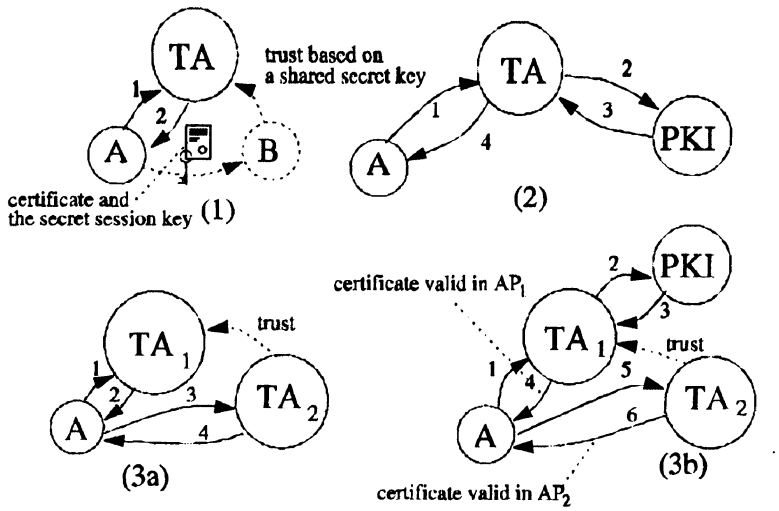

Figure 2 – Authentication Cases

a) An agent will be responsible for the access control to its resources by itself. This implies that the service providing agent has to maintain a list of authorized agents, i.e. their AID's, and operations granted to them.
b) An access to particular resources will be granted by a voucher issued either by a TA, e.g. the DF, or the service providing agent. Access to the requested resource/service will be granted in exchange for that voucher[8].
c) Access is based on mapping domain access rights appertaining to an accessing agent the rights of the local system. For the rights mapping, the resource providing agent will be responsible. Such a mapping makes the system interoperable with legacy systems with its own access control mechanism.

The approach ad a) can be denoted as a traditional approach, well known from the client/server applications. It is feasible for resources with few accessing agents as the overhead caused by the access control list maintenance increases with the number of accessing agents. The approaches ad b) and c) move the access control responsibilities to a central location, thus those methods are feasible for environments having large number of agents. Utilization of AID for decision making on access rights to services implies that the very first authentication should proceed as early as during agent's registration to an AMS.

The architecture relies on three types of vouchers:

- Time limited voucher granting access to the service for all time of its validity and bound to its holder it was issued for. This type of the vouchers will be issued by the DF according to the AP's access control policy.
- Voucher application limited, negotiable, and by the Transaction Fairness Service registered. Every application has to be mediated by the Transaction Fairness Service. This type of voucher can be issued by the service provider.
- Portable and platform independent voucher digitaly signed by a TA.

As a part of the voucher information on administrative domains, roles, and groups assigned to the agent will be included. Administrative domains are introduced to maintain the access control lists easily and are proposed to be settled hierarchically above the FIPA agent platforms.

4. CONCLUSIONS

Basic efforts aimed at achievement of FIPA agent security, which are currently being implemented within our multiagent infrastructure, are discussed in the paper. To be honest, there are additional requirements concerning the FIPA standard augmentation to support the proposed architecture - namely introduction of public-key infrastructure agent and accounting and trusted arbiter agent - which were not discussed and depart beyond the limited extent of the paper.

5. ACKNOWLEDGMENTS

This work was partially supported by the Ministry of Education of the Czech Republic under the Project LN00B096.

6. REFERENCES

1. The Foundation for Intelligent Physical Agents (FIPA). http://www.fipa.org.
2. FIPA Agent Management. Part 1 - document number FIPA00002. Foundation for Intelligent Physical Agents, 1998.
3. FIPA Agent Security Management. Part 10, Version 1.0. Foundation for Intelligent Physical Agents, 1998.
4. FIPA Agent Management Specification - document number XC00023H. Foundation for Intelligent Physical Agents, 2000.
5. FIPA Agent Message Transport Service - document number XC00067D. Foundation for Intelligent Physical Agents 2000.
4. Schneier, Bruce. Applied Cryptography. New York: John Willey & Sons, 1996.
5. Stallings, William: Cryptography and Network Security. Principles and Practice. Prentice Hall, 1999.
6. Wong, H. Chi and Sycara, Katia: Adding Security and Trust to Multi-Agent Systems. Carnegie Mellon University, Pennsilvania:1998
7. Wulf, Wm A., Wang, Chenxi, Kienzle, Darrell: A New Model of Security for Distributed Systems. Computer Science Technical Report CS-95-34, University of Virginia, Virginia: 1995.

---

[1] The reason to extend the FIPA mediator functionality instead of adding new mediators stems from the fact that the new mediators would essentially double functionality of the existing ones. The only exception is introducing of a new security agent responsible, e.g., for the PKI infrastructure maintenance.

[2] E.g. in (FIPA, 2000): "... it (DF) must strive to maintain an accurate, complete and timely list of agents ...", in the next paragraph one can read: "... the DF cannot guarantee the validity or accuracy of the information that has been registered with it ...". Some of these problems are evidently caused by lack of built-in security elements as authentication, etc.

[3] If we view a mobile agent as a service extension of the host machine (agent) and the host agent is responsible for it, i.e. the mobile agent is authenticated, trusted, etc., i.e. the mobile agent delegates its responsibility to the host agent, then everything should fit in with the paper well.

[4] It enables processing and forwarding of the message even by agents not knowing the security mechanisms.

[5] An authentication in connection with the Transaction Fairness Service, e.g., a signed message digest, can serve as an evidence about transactions, etc.

[6] In (Schneier, 1996), it is stated that symmetric algorithms are generally at least 1000 times faster than the public-key algorithms.

[7] Moreover, there are some safety reasons predestinating the public-key cryptography mainly for identity proofs and secret key exchanges, e.g., see (Schneier, 1996) and (Stallings, 1998).

[8] The voucher figures here as a digital cash.