

Supporting Security in an Electronic Market System on the Base of Web Services

Michael Christoffel, Moritz Killat

Institute for Program Structures and Data Organization
University of Karlsruhe
76128 Karlsruhe, Germany
{christof, killat}@ipd.uka.de

Abstract. Security is a precondition for the success of an electronic market system. Unless the system is able to provide some mechanism to ensure authentication, authorization, and secure data transmission, the system will have problems to be accepted by both customers and providers. In this paper, we will discuss some ideas how to add security to a distributed electronic market system that is designed platform-independently on the base of web services. We will show how we have implemented these ideas in a system for an electronic market for scientific literature.

Keywords. Security, Electronic Market, Web Services, Digital Libraries

1. Introduction

In recent years, the mention of problems in the field of computer security has become quite often in newspapers and even in the TV news. They report on dangerous viruses, computer crime, new security holes in standard software products, and network attacks that make a company's computer system collapse. Practically every Internet user feels the consequences of security lacks day after day: The masses of spam email sent every day attest a misuse of personal data.

Every computer system should be able to consider security systems in some way. This is especially true for electronic commerce systems. Hereby, customers and providers have the same intentions. Customers do not want other people monitor what they are doing, and they want their personal and financial data kept secret. Companies want to provide reliable services in order to content their customers, and, of course, they do not want that others know their company secrets.

However, the big handicap is that the Internet is not secure at all. It is relatively easy to intercept a message while it is transported through the Internet. The message can be read or manipulated without giving sender or receiver the chance of notice. It is even possible to fake complete messages.

Network security encompasses four properties:

- Authentication: Both sender and receiver of a message must be able to clearly identify themselves.
- Authorization: Each person allowed to access the system can do this only within the range of the given rights or privileges.

- Privacy: Private data must not be given in the hands of any other person without the permission of the owner.
- Integrity: Messages transmitted between sender and receiver must not be changed during transmission.

It is not possible to guarantee the latter two properties in an Internet application. However, it is possible to weaken the properties without losing the practical consequences:

- Privacy: In the case that a person's private data falls in the hands of a third person without the permission of the owner, the third person must not be able to use these data.
- Integrity: When a message has been changed during transmission, the receiver must be able to notice the change. Therefore, he/she will be able to throw the message away and ask the sender for a new transmission.

In this paper, we will show how to implement these security properties in an electronic market system, namely the UniCats system which aims to assist a market of scientific literature [3].

The work presented in this paper is supported by German Research Association (DFG) as a part of the national German research initiative "Distributed Processing and Delivery of Digital Documents (V³D²)".

We will continue as follows: In the next section, we will shortly describe the UniCats system in the stage before we added security concepts. Then we will discuss the cryptographic algorithms and standards which can be used for the solution of the security problems. In section 4, we will introduce the main ideas of our security improvements, and in section 5, we will describe how we have realized these ideas. In section 6, we will shortly mention some related approaches. We will conclude this paper with a summary of the main points of this work and a discussion of future work.

2. The UniCats System

Background of the UniCats¹ approach is the investigation of information markets. Information has an important role in the modern society, where many professions depend on a steady supply with actual information.

The application domain are markets of scientific literature. These markets have received much attention in the last years due to the success of the Internet, which allows publishing houses, booksellers, and libraries to offer their goods and services world-wide. Additionally, we can observe the growth of completely new kinds of information providers such as delivery services, bibliographic databases, and citation servers. At the same time, the amount of scientific literature demanded by students and scientists at university and research institutions has increased rapidly.

However, the customers in these markets are confronted with the problem of information overload. Often they are not able to survey the multitude of information

¹ a Universal Integration of Catalogues based on an Agent-supported Trading and Wrapping System

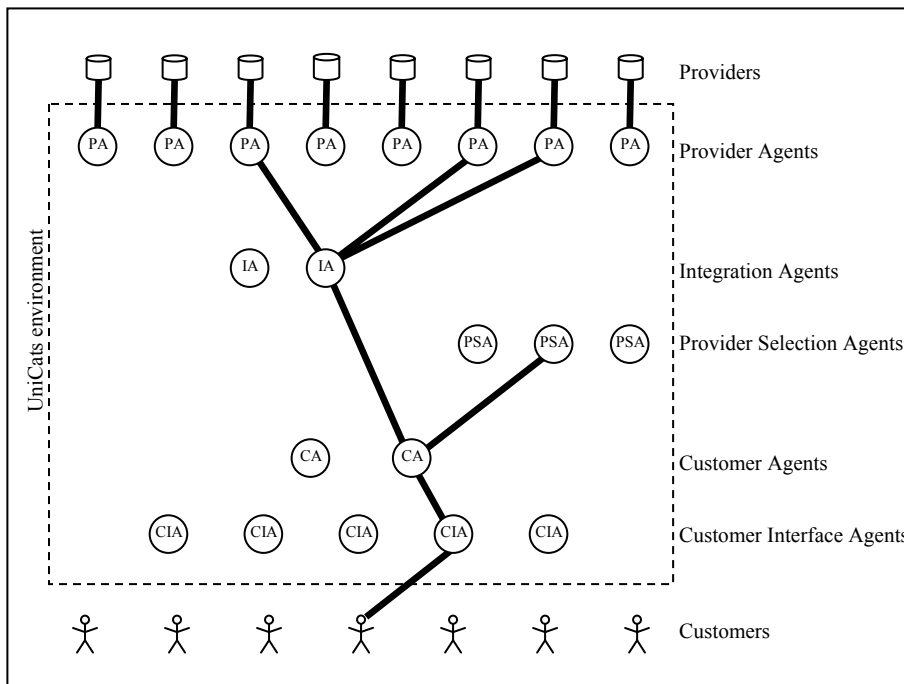


Figure 1. The UniCats environment

sources available or to find the providers most appropriate for their demands. In order to proceed with their search task, they have to apply several services in a sequence, evaluate and compose search results, and find ways of document delivery. Doing all this manually can be expensive in time and money.

The UniCats approach intends to develop a system for the integration of information services and sources, handling heterogeneity and distribution, automating search processes, and providing the customers in these markets a uniform access to the information available [4].

We have developed an agent-oriented platform for the support of an open information market, which we refer as the UniCats environment. In our market model, both customers and providers are autonomous and independent and can enter and leave the market on their own decision without further notice. A discussion of this market-oriented approach can be found in [1]. Although we have considered only markets of scientific literature until now, both the market model and the agent platform are general enough so that they could also be applied to other scenarios and application domains.

Figure 1 contains a simplified view on the UniCats environment, showing the most important agent types. Customer Interface Agents (CIA) are the connection of the customers to the system. This connection is established by a set of user interfaces that can be customized to the customers' personal preferences. Different user interfaces can be used in different situations. So a customer might have a favorite user interface that uses all the capabilities of his/her desktop computer, but

will use a different user interface when he/she will have to contact the UniCats system through a mobile device during a business travel.

Customer Interface Agents are connected to customer agents (CA) that act as the representatives of the customers in the system and provide a personal workplace for each customer. In order to perform queries in behalf of the customers, Customer Agents can ask a Provider Selection Agent (PSA) for recommendations on those providers, which are most appropriate for the customer's demand.

A Customer Agent sends a query together with a list of the selected providers to an Integration Agent (IA). The Integration Agent sends the query in parallel to the Provider Agents (PA) that act as the representatives of the providers. The Provider Agents translate the incoming query into the native protocol of the provider and re-translates the delivered results into the common protocol. The Integration Agent collects the incoming results from the different information sources and integrates them to a uniform result list. The final result list is sent back to the Customer Agent, which can use the results for the further task execution or present the results to the customer with the help of the Customer Interface Agent.

It is important to consider that this scenario is only an example of possible interactions. A more complex scenario may contain several customers who operate with the system at the same time, require the combination of different queries including order and delivery, and involve many different agents.

In addition to the agents shown in Figure 1, we have implemented some more agents: Customer Organization Agents (COA) represent the interests of customer organizations (such as universities, research institutes, and companies) and their members. Billing Agents (BA) manage financial transactions among the agents of the environment. External Payment Agents (EPA) provide interfaces to financial institutions for the access to external accounts and methods of digital payments. Agent Naming Agents (ANA) and Group Naming Agents (GNA) provide a naming service for agents and groups of agents. System Administration Agents (SAA) monitor the entire environment (or a part of the environment) and assist the system administrator in case of a failure.

It is possible to add new agent types that can be derived from existing agent types or from the agent basic class.

For a more detailed description of the UniCats environment see [3].

We implemented the UniCats environment using Java programming language. This brings the advantage that our agents can run platform-independent on most computers. There is also a large set of free tools and class libraries available. However, the development of agents does not depend on the chosen programming language. It is also possible to implement agents using other platforms and languages, and these agents will work together in one environment. We tested this with a sample environment encompassing different hardware platforms and agents written in seven different programming languages.

The UniCats environment is a distributed system that can encompass several computer nodes. Each computer node can hold one or more UniCats communities.

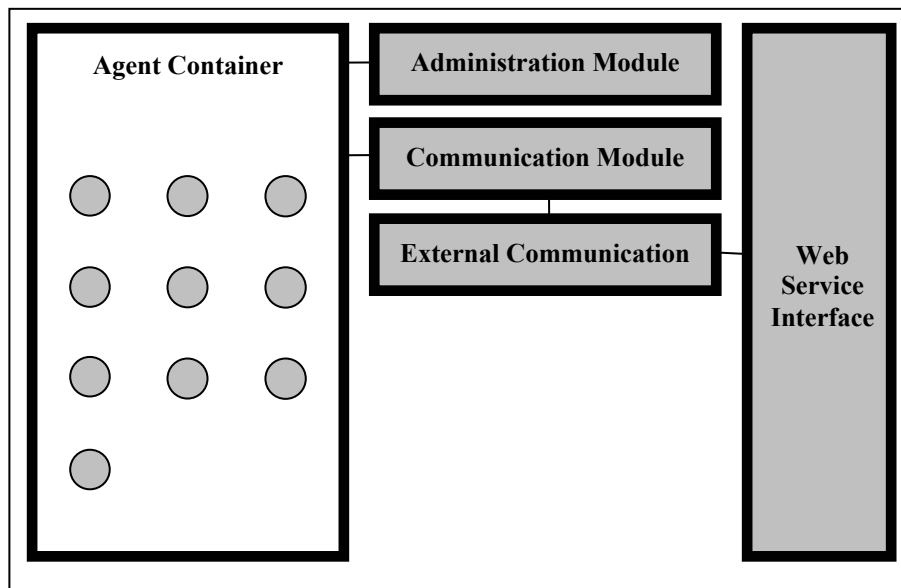


Figure 2. The UniCats community (without security extensions)

Figure 2 shows the basic structure of a UniCats community. The community consists of an agent container and three modules: Administration Module, Communication Module and External Communication Module. The agent container can hold any number of UniCats agents, sharing resources. Agents can be added and deleted at runtime. It is also possible for agents to migrate from one agent community to another.

The administration module is the main module of the community. It is responsible for the initialization of the community and controls startup and shutdown of the agents. The communication module is responsible for the communication of all agents in the community and manages outgoing and incoming messages. There are four different ways of message interchange among the agents:

- Agent communication is used between two agents.
- Group communications is used between an agent and a group of agents.
- Community communication is used between an agent and all the agents in a community.
- System communication is used between an agents and a community itself.

While messages directed to an agent inside the own community are forwarded to this agent on a direct way, the Communication Module delegates messages that are supposed to be delivered outside the community to the External Communication Module. Similarly, the External Communication Module receives all messages coming from outside the community and forwards them to the Communication Module.

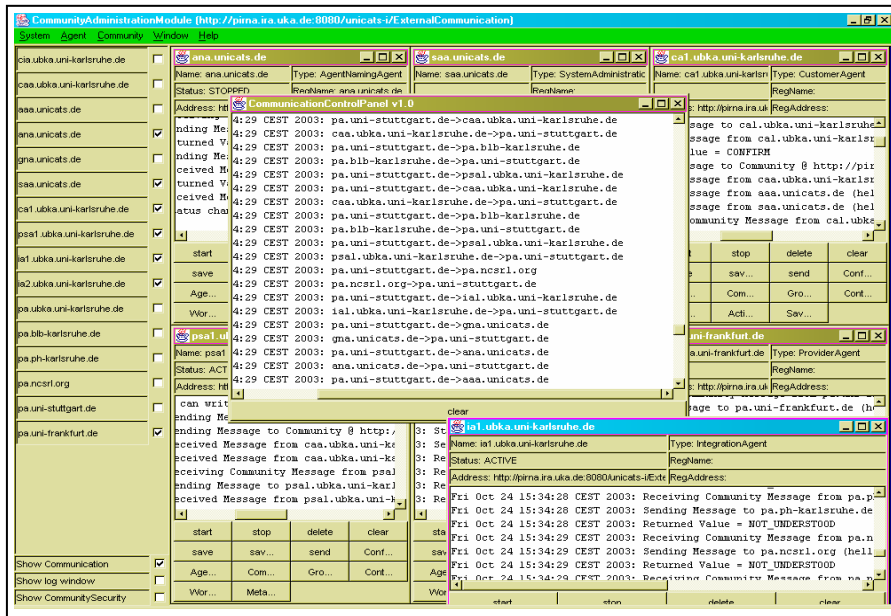


Figure 3. Administration control panel of a UniCats environment

Each module and each agent has a graphical interface, the control panel. The control panels are used to survey and administer the agents of one community. They are hierarchically structured with the administration control panel as the parent frame (Figure 3). In addition to the direct control through the control panel, it is also possible to configure the agents and the community with human-readable configuration files. Important commands can also be applied through a text-based command prompt.

Any communication between different UniCats communities is operated by web services. Each community has a Web Service Interface which is controlled by the External Communication Module. This way, every transmitted message – including all parameters – is automatically converted into an XML document which is delivered to the receiver web service using standard Internet protocols. The use of web services as transport layer is the main reason for the ability of the UniCats system to build cross-platform applications. Another advantage is that we can overcome the firewall problem. While many network administrators close Internet ports for safety reasons, UniCats is not touched by this, because the web service communication uses only those standard ports that are accessible at every system.

However, we have seen that the Internet does not provide privacy. Messages transmitting as XML documents through the Internet are free for unauthorized reading and manipulation. Moreover, the openness of the UniCats environment allows agents and market participants to penetrate uncontrolled into the system. If UniCats should really be recognized as a platform for electronic commerce, we need

a solution for these security problems. Before we will present the solution we have developed, we will first introduce some basics in cryptography.

3. Cryptography

When a message, which content is supposed to be kept secret, has to be transmitted through an unsafe medium (such as the Internet), then it is not possible to exclude the danger that this message comes in the hand of a third person. A solution for this problem is to use encryption: The sender transforms the message into a cipher text, and only a receiver who knows the right key can re-transform the message into a readable form. For all other persons the cipher text is useless.

The easiest way to establish encryption is symmetric encryption, which uses only one key for both encryption and decryption. Principally, symmetric encryption bases on a reversible XOR operation between message and key. One of the most popular symmetric encryption algorithms is DES (Data Encryption Standard), which has been standardized in the year 1977 and uses a 56 bit key. However, due to the relatively short key length and the increased power of the contemporary computer systems, DES can be defeated by a brute-force attack. Therefore, a modified algorithm has been published under the name 3DES (Triple DES), which uses a key length of 112 or even 168 bit. A possible successor is AES (Advanced Encryption Standard), which has been standardized in the year 2000 and can support key length up to 256 bit.

However, symmetric encryption has two major disadvantages. First, it is necessary to have a separate key for each pair of sender and receiver. Second, since both sender and receiver must know the symmetric key, this key has to be transmitted over the network at least one time, giving an attacker the chance of interception.

Both disadvantages can be solved using asymmetric encryption. Here two keys are involved. A message encrypted with the first key can be decrypted with the second key and vice versa. Usually one key is kept secret (private key), while the second key is published (public key). When a sender intends to send a message to a receiver, the sender uses the public key of the receiver for encryption. This way, only the receiver who holds the corresponding private key can decrypt the message. The most popular asymmetric key algorithm is RSA (Rivest Shamir Adleman) from the year 1978, which bases on prime factors. Even today, RSA with 2048 bit key length is supposed to be very safe.

The drawback of asymmetric encryption is performance. Symmetric key algorithms are faster than asymmetric key algorithms, to be exact by the factor 100-1000. Therefore, practical applications use a combined procedure, using asymmetric encryption for the exchange of a symmetric key, and then use symmetric encryption for the message itself.

An alternative way for key exchange provides the Diffie Hellman algorithm from the year 1976, which bases on the problem of discrete logarithms. The Diffie Hellman algorithm enables two communication partners to arrange for a common symmetric key by an iterative protocol. With the Diffie Hellman algorithm it is not necessary to transmit the symmetric key at all, and this algorithm is much faster than

asymmetric key algorithms. However, the Diffie Hellman has another disadvantage: it does not stand a man-in-the-middle attack. For an attacker, it is possible to interrupt the communication between the two original communication partners and make both sender and receiver arrange common symmetric keys with him/her.

Another application for asymmetric encryption are digital signatures. For digital signatures, the sender encrypts a message with his/her private key. The receiver tries to decrypt the message with the public key of the sender. If the receiver succeeds in the decryption, then this is a proof that both keys belong together. The message really comes from the owner or the public key and not from somebody else.

However, there is one difficulty left. How can a communication partner be sure that a public key really belongs to a definite person and not to somebody else, maybe an attacker?

The solution for this provide certificates. A certificate holds the name and the public key of the sender and is digitally signed by a trusted third party. However, the signature of a trusted party only proves that the certificate has been in the hands of this third party. It does not automatically prove the correctness of the certificate (or any other document). For example, an attacker could have caught the original certificate and simply exchanged the contained public key by its own public key, then sends the certificate to the trustful receiver.

In order to prevent such manipulations, it is possible to apply one-way hash functions. The most popular algorithms for the calculation of such hash functions are MD5 (message digest 5) from the year 1992 and SHA (Secure Hash Algorithm) which has been published in the year 1995. They use hash values of a length of 128 and 160 bit, respectively.

In order to give a digital signature that cannot be manipulated, the author calculates the hash value of the document that is to be signed, encrypts the hash value with his/her private key, and adds the encrypted hash value to the document. The receiver splits and decrypts the added hash value, then calculates the hash value of the message on his/her own. If both values match, the receiver can be sure that the message comes from the signing sender and has not been manipulated.

4. Security Concept

In section 1, we have exposed authentication, authorization, privacy, and integrity as the challenges for a security concept. For the UniCats system as it has been presented in section 2, we see the following four starting points for a security improvement:

- Authentication of customers: The identity of a customer must be assured.
- Authorization of customers: Customers using the system may have different rights. E.g., in a university library system, faculty members may have more rights than students, and these have more rights than external users.
- Authentication of agents: The identity of an agent must be assured.
- Secure communication channels: It must be possible to encrypt messages transmitted between agents so that only the intended receiver of the message is able to understand the message and manipulations of the message can be detected.

Authentication and authorization is managed by customer passports. The customer passport verifies that a specific customer has successfully passed login procedure and confirms the rights of the customer in the use of the system. The customer passport accompanies each query and order of a specific customer. After a customer has left the system, all instances of the passport should have been deleted. If the customer logs in again, a new customer passport is created. In addition to that, each customer passport is equipped with an expiration date and becomes invalid after a period of time.

Customer passports are issued by a new agent type, the Customer Authentication Agent (CAA). During login procedure, a Customer Interface Agent contacts the corresponding Customer Authentication Agent and sends the customer's account name and password (in an encrypted form). The Customer Authentication Agent checks whether the customer is registered in its database and the passport matches. Either it answers with a new issued customer passport, which includes the rights of the customer, or it sends an error message, which causes the Customer Interface Agent to refuse the entrance to the system. There is also the possibility to issue an anonymous guest passport; however, this passport has very limited rights.

The authentication of agents can be established very similar to the authentication of customers. An agent that is about to enter a business relation to another agent can ask for this agent's agent passport. The agent passport holds information about the name, current address, and type of an agent. If an agent moves its location or the expiration date is reached, a new agent passport must be issued. Agent passports are issued by an Agent Authentication Agent (AAA).

Customer passports and agent passports are signed with the digital signature of the issuing agent. A hash value is used to detect manipulations.

Since all communication inside a community is done by direct procedure calls, secure communication is only necessary for external communication, when messages are exchanged by means of web services. For performance reasons, we use symmetric encryption and create the symmetric key just in time using the Diffie Hellman algorithm. The protocol we use is the same as the protocol used for secure communication in SSL, so it is likely that we can provide the same level of security.

For secure communication, the UniCats community has been extended and a new module has been added, the Secure Communication Module. This module offers message encryption and decryption by means of a symmetric key negotiated with the community of the communication partner. When an external community is contacted for the first time, the Secure Communication Module invokes the Diffie Hellman protocol in order to create a new symmetric key. A symmetric key is only valid for a pair of communities and never transmitted through the network.

In order to prevent a man-in-the-middle attack, each community should have a communication certificate. Before the negotiation for a symmetric key starts, both communication partners exchange their communication certificates in order to prove the identity of the other communication partner. In contrast to a passport, the certificate also holds the public key of the owner of the certificate. The private key associated with the community certificate is never transmitted. Unless a community owns a certificate, it can not provide secure communication.

Encrypted communication is not necessary in each case. An agent can decide in each single case whether to apply secure communication or regular communication.

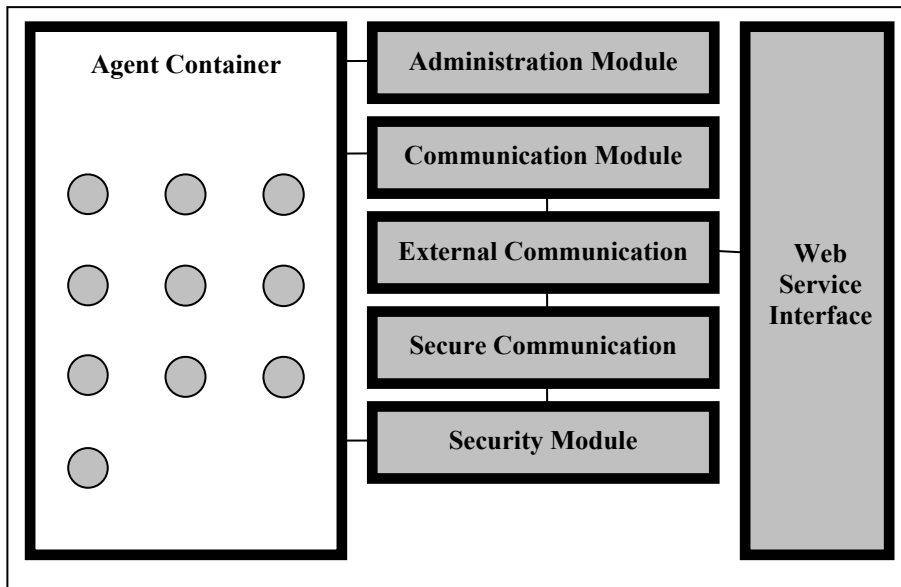


Figure 4. The UniCats community with security extensions

Regular communication is much faster than secure communication and always available.

We have seen that each community needs a community certificate in order to provide secure communication. A similar certificate is needed by the Customer Authentication Agent and the Agent Authentication Agent in order to sign customer and agent passports. These certificates are issued by a new community module, the Security Module. The Security Module holds a permanent and unique certificate, the community certificate. All certificates issued by a Security Module are signed with the community module.

If a community does not own a community certificate, it can ask another community to issue a certificate. This procedure leads to a hierarchy of trusted communities. The private keys associated with the certificates are always kept secret.

Each Security Module holds a list of trusted certificates. Hence, a Security Module is able to prove certificates and passports without having the need to contact the issuer. However, an agent may decide to contact the issuer instead or in addition to proving a passport by the local Security Module, in order to become sure that the passport has not been revoked.

Figure 4 shows the structure of the UniCats community with the two new modules.

5. Realization

The security extensions have been implemented using Java programming language in line with the rest of the UniCats community. However, the security extensions are principally independent from the applied programming language, using only standard protocols and algorithms. For example, secure communication can be established between an agent implemented in pure Java and a second agent implemented in C#.net.

The new agent types Customer Authentication Agent and Agent Authentication Agent are implemented as extensions of the UniCats standard agent class and are therefore compatible with the rest of the system. Both agent types can be controlled and maintained by an agent control panel.

The Customer Authentication Agent holds an internal database with the registered customers and their passports (in a one-way encrypted form). New customers can be entered by a registration procedure provided by the Customer Interface Agent. Of course, customers can edit their entry, change passports, or delete their registration. Another possibility is to provide a connection to an external database, containing, e.g., a list of all members of a university and their position. The administrator of the agent can edit the registry of the registered customers at any time, delete a customer, whose registration has been revoked, or change the rights of a customer. The Customer Authentication Agent also holds a revocation list, publishing those customers whose registration has been revoked, but might have a passport still valid.

The Agent Authentication Agent is very similar to the Customer Authentication Agent. In order to decide whether a passport is issued for a specific agent, the issuing agent checks whether the other agent has a registered type. The Agent Authentication Agent holds a revocation list, too.

The Security Module issues certificates to the own Secure Communication Module, the two types of authorization agents, and other communities. Issuing a certificate to another community gives this other community the possibility to issue certificates on its own and can be a source for misuse. Because of this, a community needs the confirmation of a human administrator to issue a certificate to another community. In addition to this, each Security Module holds a revocation list of invalidated certificates, which is propagated among the trusted communities.

Each certificate saves the whole history of issuers, i.e., not only the community which issued the certificate, but also the community which issued the certificate of the issuer. This eases the proof of a certificate (or a passport) inside the Security Module, because the Security Module can trust all certificates in this sequence as long as it can trust one of the certificates. Moreover, this also works the other way: If a Security Module cannot trust one certificate (e.g., because this certificate has been revoked), it can also invalidate the certificates following in the hierarchy.

For the Security Module, a separate control panel (the security control panel) has been created and integrated in the frame of the administration control panel.

6. Related Work

The most popular protocol for providing secure transmissions through the Internet is the Secure Socket Layer SSL [6]. SSL bases on both certificates and data encryption, using a symmetric key that is created with the Diffie Hellman Algorithm. SSL has two major disadvantages in regard of web services: Each messages transfer has to be encrypted, even if this is not necessary, e.g., if the content of the message does not need to be kept secret. The second disadvantage is that the message is encrypted completely. For a web service application, it would be more appropriate to encrypt only some parts of the messages, while other parts such as the address or transport information should be readable by everyone.

For our security enhancement, we used the same protocols as SSL. Since we implemented the encryption in the application level, we could avoid its disadvantages.

Another important approach in the encryption of XML documents comes from the World Wide Web Consortium W3C [8]. XML Encryption replaces any part of the XML tree by an encrypted element that is enclosed in an EncryptedData tag. XML Signature supports the transport of digital signatures. For a detailed description see [5]. The XML Key Management Specification [W3C-XK] supports the access to a public key infrastructure and provides a mechanism for the registration of public keys independent from the actual implementation of the certificates (compare [7]).

7. Conclusion

In this paper, we have presented the idea and the realization of providing security to a platform for an electronic market in the field of scientific literature. The security enhancement encompasses four objectives: authentication, authorization, privacy, and integrity. The main parts of our solution are a combination of mutual exchange of signed certificates and (lightweight) passports and the encrypted communication on the base of web services.

The main advantage of our approach is the security mechanism may only be used when it is necessary. Secure communication is only applied when private data of a customer or agent are transmitted; an exchange of agent passports or communication certificates only takes place the first time when two agents or communities come into contact and the authentication is mandatory. In general, the decision whether security mechanisms are applied or not is left to the agents.

There are several extensions worth to be considered in the future:

- The type checking used by the Agent Authentication Agents and the Security Module for their decision, whether to issue an agent passport or a certificate for an authentication agent, is not safe. As long as all source code for all agents is freely available, an attacker could simply camouflage his/her own agent with a standard agent type, calculating the necessary hash value on his/her own. A solution for this would be to register all agents by a central authority and include a unique agent identifier in the source code of an agent implementation.

- We have applied authentication and secure communication only inside the UniCats environment. There is no guarantee about the connection to customer and providers. However, these connections are very individual and depend on the preferences of customers and providers. It is not possible to create a general solution.
- In order to have a proof about the business transactions performed within the environment, it would be necessary to have a log of all these transactions. Agents do have a private log, but this could be faked very easily. So there should be a separate trusted log instance, where all transactions are reported. However, since all messages had to be sent twice, such log instance would cause a sincere increase of network traffic. Moreover, monitoring all steps of a customer would also be a contraction to the goal of data privacy.

References

- [1] M. Christoffel: *Information Integration as a Matter of Market Agents*. In: Proceedings of the 5th International Conference on Electronic Commerce Research, Montréal, 2002
- [2] M. Christoffel, G. Wojke, M. Gensthaler: *How Many Small Libraries Can Be a Large Library*. In: Proceedings of the 5th Russian Conference on Digital Libraries. St. Petersburg, 2003
- [3] M. Christoffel, B. Schmitt, S. Pulkowski, P. Lockemann: *Electronic Commerce: The Roadmap for University Libraries and Their Members to Survive in the Information Jungle*. In: ACM SIGMOD Record 27(4), 1998
- [4] M. Christoffel, B. Schmitt, S. Pulkowski, P. Lockemann, C. Schütte: *The UniCats Approach: New Management for Books in the Information Market*. In: Proceedings of the International Conference IuK99 – Dynamic Documents, Jena, 1999
- [5] M. Mactaggart: *Enabling XML security*. <http://www-106.ibm.com/developerworks/xml/library/s-xmlsec.html/index.html>
- [6] Netscape: *SSL 3.0 Specification*. <http://wp.netscape.com/eng/ssl3/>
- [7] M. Verma: *XML Security: The XML Key Management Specification*. <http://www-106.ibm.com/developerworks/xml/library/x-seclay3/>
- [8] World Wide Web Consortium: *Homepage of the XML Encryption Workgroup*. <http://www.w3.org/Encryption/2001/>
- [9] World Wide Web Consortium: *Homepage of the XML Key Management Specification*. <http://www.w3.org/2001/XKMS/>