An Approach for Mobile Agent Security and Fault Tolerance using Distributed Transactions

Hartmut Vogler, Thomas Kunkelmann, Marie-Louise Moschgath
Darmstadt University of Technology
Information Technology Transfer Office
Alexanderstr. 10
D-64283 Darmstadt, Germany
{vogler, kunkel, malu}@ito.th-darmstadt.de

Abstract

Mobile agents are no longer a theoretical issue since different architectures for their realization have been proposed. With the increasing market of electronic commerce it becomes an interesting aspect to use autonomous mobile agents for electronic business transactions. Being involved in money transactions, supplementary security features for mobile agent systems have to be ensured.

In this paper we present an architecture for a mobile agent system which guarantees security for the host as well as security for the agent. This architecture additionally offers fault tolerance for the whole agent system at a high level. To handle these issues for mobile agents we use various encryption mechanisms and we apply a novel method for mobile agent systems by using distributed transactions processing based on the OMG Object Transaction Service in our architecture. With this security architecture an agent will be enabled to do money transactions.

Keywords: mobile agents, security, fault tolerance, electronic commerce, distributed transactions, CORBA, OTS

1 Introduction

Mobile agents [HCK95] offer a new possibility for the development of applications in distributed systems. Mobile agents are no longer a theoretical issue since different architectures for their implementations have been proposed. Research institutes as well as companies develop high-quality prototype systems for mobile agents, e.g. *ARA* [PeSt97], *Mole* [SBH96] or *Telescript* [Whi94]. Yet, these agent systems typically satisfy not all requirements mobile agents need for a full environment. However, these systems are open for further extensions to include additional characteristics.

For new and useful applications, mobile agent systems must also provide additional features to handle the security for the agent and the host [FGS96], fault tolerance and electronic commerce [KaWh96]. Only with those security guarantees and extensions, mobile agents will be used in a wide range of applications.

In this paper we describe a security architecture for mobile agents that can be used on top of existing systems. To handle the open issues for mobile agents we adopt a different method by using *distributed transactions* in our architecture.

The core of our secure agent system builds an instance called *trust service* and a special protocol for the agent migration, which is a combination of data logging, encryption mechanisms and distributed transactions. For the agent we offer to be also part of the electronic market place due to a bank service, based on the concept of *First Virtual* [FV96] with a special extension for mobile agents.

In section 2 we discuss the importance of a reliable agent transfer. Based on this analysis we explain our architecture of a secure mobile agent system, which introduces the concept of distributed transactions, and discuss its benefits. After describing our realization using CORBA and OTS in Section 4 we conclude the paper with a summary and an outlook on our future work.

2 Reliable Agent Transfer

Fault tolerance for mobile agent systems is an unsolved topic, to which more importance should be attached. Besides the security problems by intended attacks it is very important to realize that an agent can simply get lost by errors of the network or the hosts. If an agent itinerates autonomously in the network, there is no instance which can guarantee that the agent reaches the next host correctly and won't get lost. For example, when using SMTP

(E-mail) as a transport medium there are no mechanisms for the reliability of the transport or any recovery handling in case of errors. There are even no standardized mechanisms to detect this fault situation.

Besides the network errors, fault situations also can be originated by problems or a breakdown of the host. In case of such a failure, there have to be suitable recovery mechanisms for the agent.

Based on network or host errors an agent can also be duplicated. Such an inconsistent state of the agent system can cause significant problems and must be avoided.

2.1 Classes of fault semantics

The problem of fault tolerance for agents is an application level topic. After a reliable transfer an agent has to be initiated at the new host. This offers various fault situations, which can easily cause an inconsistent state for the whole agent system.

The transfer of an agent can be compared with a *remote procedure call* (RPC), which guarantees the *exactly-once-semantic* (only-once-type-2-semantic) [MüSc92].

In the context of an agent transfer the other classes of fault semantic aren't sufficient. An agent transfer with the *maybe-semantic* would guarantee that the agent is transferred only once, but we don't have the guarantee that the agent is received and initiated correctly at the new host. In the class of the *at-least-once-semantic* we have the guarantee that the agent is received and initiated correctly by the new host. However we can not avoid that the agent may be duplicated if any communication errors occur and the agent has to be resent. The *at-most-once-semantic* (only-once-type-1) guarantees the atomicity of the agent transfer. In this class the agent will either be transferred correctly or, in case of a fault, an error message will be sent to the sending host, and no further action takes place at the receiving host. Even this fault semantic isn't sufficient for an agent transfer due to a possible host crash. Only the *exactly-once-semantic* guarantees a consistent recovery after a host crash, so an agent will always be transferred exactly once and will not get lost or duplicated. This can only be achieved with persistent storage and a protocol for distributed transactions.

After this analysis we see that only *distributed transaction processing* (DTP) offers the possibility to control the state of a mobile agent system and also offers the scalability of the system to involve further parties.

2.2 Introduction in DTP

The usage of *transactions* [BeNe96] is a very popular concept for the management of large data collections. Transactions guarantee the consistency of data records when multiple users or processes perform concurrent operations on them. In general, the properties of transactions are known as the *ACID* properties (**A**tomicity, an indivisible set of operations; **C**onsistency is guaranteed for the data; Isolation of parallel data access in different transactions; Durability, the events are stored permanently or completely rejected) [GrRe93].

The access of distributed resources, e.g. databases on different computers, within a transaction is called a distributed transaction. For committing the result, the peers involved in a distributed transaction usually communicate via the *two-phase-commit* protocol (2PC), see Figure 1. The initiator of a transaction takes the role of the coordinator, which in the first phase collects the votes about the result of the transaction from the different partners. In the second phase it transmits the result (*commit*: make the results of the transaction permanent; or *rollback*: discard all changes) to the other partners which subsequently confirm the receipt. The 2PC thus is quite robust for the communication in distributed systems. In distributed systems transactions must be used through standardized procedures and protocols due to the heterogeneity of the systems.

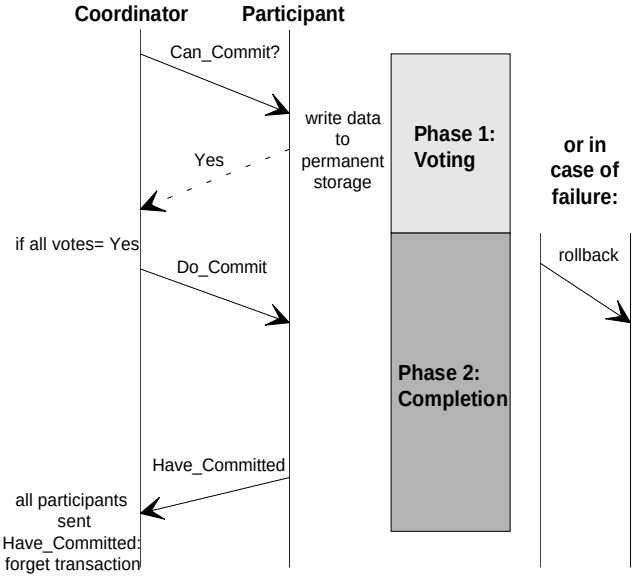

Figure 1: Tow Phase Commit Protocol (2PC)

In the last few years several service platforms (middleware) came up for communication in heterogeneous distributed systems, which have been standardized now. One of the most important platforms for distributed application development is *CORBA* (Common Object Request Broker Architecture) [OMG95] of the *Object Management Group* (OMG). The OMG has specified several object services, like naming service and timing service, for CORBA. One of these services is the *Objec*t Transaction Service (OTS) [OMG94] for object-oriented distributed transaction processing.

Nowadays the concept of transaction processing is used - apart from the "classical" database applications - in various applications, e.g. for the management of large distributed systems [Vog96] [MaHa96]. There also exist first experiences and implementations using transaction processing on the Internet [DEC96a] [LEK97], which offer a wide range of opportunities to build new secure and fault tolerant applications. Furthermore it is important to recognize that DTP systems are not monolithic, rather they are open by standardized protocols.

3 Architecture of a secure agent system

In the previous section we outlined the importance of transactions for a reliable agent transfer. However, the usage of distributed transactions and the 2PC protocol contains some drawbacks like the bad performance of the 2PC protocol due to its blocking characteristics. DTP also introduces implementation overhead and additional software components. Analyzing these pros and cons of DTP in the context of mobile agents we came to the conclusion that the benefit of distributed transactions should be used for additional valuable features of an agent system. Therefore we build an DTP-based enhancement of existing systems for host security as well as agent security, which additionally allows agent management and control. With our system we also equip an agent for electronic commerce.

3.1 Trust Between Host and Agent

Analyzing the situation in the mobile agent paradigm we came to the conclusion that it is not possible to achieve full security with complete functionality for the agent without trusting each other. By aid of a third party, which has information about all instances of the closed system, a kind of trusted situation or contract can be achieved. In our architecture we call this instance trust service. This trust service is the core of our architecture for agent security and fault tolerance. On the one hand, our concept of a trust service is based on the concept of Kerberos [NeTs94] in the way that we also use a trusted third party for a session key generation and distribution. On the other hand it is important to see the difference between our trust service and well-known security services like the security service of the *Distributed Computing Environment* (DCE) [Hu95]. Our trust service is an extension to handle the specific problems of mobile agents and is based on common security concepts and services.

Besides a special security protocol, which is explained in the following section, our trust service logs data about the agents and the hosts. These data include a record with the agent's identity, the host ID and the time interval of the visit. The trust service and the protocol additionally guarantee the correctness and consistency of these data. With these data we have enough information about the agent and the host so that in case of a breach of trust we can ascertain the originator and call him to account. Our system also offers the basis for agent control and management.

3.2 Protocol Steps

In this section we describe our protocol for the agent migration, which is a combination of data logging, encryption mechanisms and DTP. To use PGP [Gar94] in our system we assume that every participant in the agent system possesses a certified public key [Kal93]. We also assume that every instance, hosts as well as users of the agent system, are registered at the trust service. This registration must be associated with a legally binding contract in order to achieve a trusted starting point.

In conjunction with the registration a user announces to the bank that he will participate in the electronic cash system and transmits via a secure channel his bank account number and his credit card number. The electronic cash system establishes a virtual account and notifies the user about the virtual account number. All purchases with this virtual account number have to be confirmed by the owner before the credit card will be charged. This concept introduced by First Virtual [FV96], with freely accessible data for the payment, seems to be the suitable mechanism for mobile agents.

Other electronic cash systems are based either on a secret (credit card number) like the iKP (Internet Keyed Payment Protocol) of IBM [IBM96] or on electronic proxies for money (electronic coins or coupons) like DigiCash [Dig96] or Millicent [DEC96b]. In either of these cases the agent has to carry out a secret information. So these solutions are not suitable for mobile agents.

The following items describe the life-cycle of an agent in our architecture:

- The originator announces a new agent to the trust service. The trust service generates a unique ID for the agent, which is used in conjunction with the registered user and which is sent back. This communication is protected by public key encryption.
- When the agent is initiated at the host of its originator, it looks for a new target host with aid of a special trader for agents. To find a suitable new host this trader must have additional information concerning the resources, environment conditions and services at the offered host.
- When the contract between the target host and the agent is negotiated, a copy of the agent will be transferred to the new host. To achieve security we use two different mechanisms. Distributed transaction processing guarantees a consistent state of the whole system during transport. Encryption preserves the system

from attacks in an unreliable network. Figure 2 illustrates the different protocol steps of an agent transfer:

1. The originator host (H_O) demarcates the begin of a distributed transaction (*tx_begin*), in which the trust service and the target host (H_T) will be involved. This implies that all action and commands will be executed in a transactional context.
2. The originator host requests a *session key* (S) [Sch96] for the secure transfer of the agent to the target host.
3. The trust service generates a session key and propagates it to the originator and the target host by using public key protocol (PP).
4. The originator host transfers a copy of the agent, encrypted with the session key, to the target host. The open design of our architecture allows to use various mechanisms or protocols for the agent transfer. So we can use E-mail [Bor94], HTTP like it is used in [LDD95] or the suggested *Agent Transfer Protocol* [Lan96].
5. After decrypting, the target host initializes the copy of the agent and acknowledges the receipt.
6. The originator host initiates the 2PC protocol to conclude the transaction. A successful conclusion of the transaction implies that the results, i.e. deletion of the old agent, start of the new agent and update of data at the trust service, are made permanently visible. With the end of the transaction the session key is invalidated.

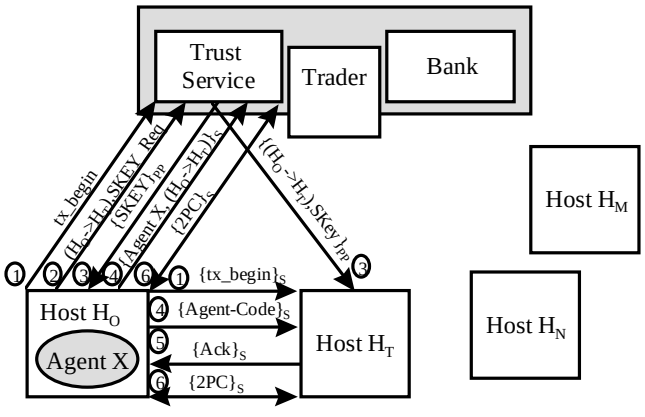

Figure 2: Protocol steps of an agent transfer

- Being transferred successfully the agent can be involved in a business transaction either as a purchaser or as a vendor. After an agreement is arranged, the purchaser gives his virtual account number to the vendor, who contacts the bank and transmits the virtual account number together with information concerning the business, the identifiers of both parties and the host. The bank checks the consistency of the information, acknowledges the validity of the business transaction and transfers the money to the vendor's account.
- For a correct termination of an agent an appropriate message must be sent to the trust service, which logs this information. From this moment the agent is invalid.

3.3 Benefits of our architecture

We now emphasize the benefits that our architecture brings into a mobile agent system.

Security for the host

With our architecture we guarantee that an agent is only introduced in the system by a trusted user and can only itinerate on a chain of trusted hosts. We log all the relevant data about the visiting agent, like information about the owner and the life cycle of the agent. In case of a breach of trust by the agent the originator of the agent can be ascertained and called to account for the actions of his agent. With our architecture we achieve an indirect security for the host without any restriction to the capabilities of the agent in contrast to the concept of "safe" languages like *Safe-TCL* [BoRo93] or *Safe-Python* [Pyth94].

Security for the agent

Our guarantee for the security of the agent include various aspects. We guarantee that besides the scope of trusted hosts no other instance has access to the agent due to the encryption methods used. So no untrusted third party can copy, modify, destroy or rob the agent. If one of the trusted host attacks the agent, the logging facilities of our system can trace back this breach of trust and similar to the host security case, call the responsible person to account. This indirect security for the agent is also achieved without any restrictions to its capabilities.

Fault tolerance and reliable agent transfer

The usage of distributed transaction processing in our protocol offers a new opportunity for the fault tolerance of an agent system. This new technique for the agent migration avoids the duplication of an agent as well as its loss. During the critical phase of the transport of an agent the 2PC protocol guarantees the preservation of a consistent state of the whole system. In case of all error situations and host crashes the recovery and rollback mechanisms of the DTP system cater for a reliable and consistent resumption of the agent transport.

Electronic commerce enhancement for mobile agents

Our architecture offers an agent to cope with money transactions based on the concept of First Virtual. Addi-

tionally, the bank service can check the consistency of a money transaction by aid of the logging information. It can be validated that an agent involved in the business actually resides at the right host, from where the money transaction was announced to the bank. In case of an error or cheating of one party this can be detected an the money transaction will be refused. It is then up to the vendor and the trust service to initiate further action.

Agent control and management

With the information about the agent and its itinerary at the trust service, suitable control and management functions can be implemented. The owner of an agent can every time locate it and give new instructions to it. Also a control of its lifetime and its goals can be achieved. The trust service can additionally offer to store a copy of the agent in the case that a host does not have an persistent storage or recovery mechanisms.

4 Realization with CORBA and OTS

After the description of the protocol steps and the components of our architecture we give in this section a description about the realization. The communication is based on CORBA with its object service for transactions (OTS).

4.1 OTS Integration

It is important to notice that our system is an enhancement on top of existing mobile agent systems. The communication of our system is based on CORBA calls, while the agent migration to a new host will be done with the specific protocol of the mobile agent system.

We focus in our description on the usage of the OMG object transaction service, which provides the necessary operations

- to control the context and the duration of a transaction
- for the participation of multiple objects in a single transaction
- to combine internal changes of object states within a transaction
- for the coordination of the 2PC protocol at the end of a transaction

Figure 3 shows the coherence of the different components and objects of OTS.

- An object whose methods can be called in a transactional context is called a **transactional object (TO)**. A TO is characterized by including some persistent data or pointers to persistent data, which can be modified by its methods.

  A call of a TO need not be transactional, even if the call is within the context of a transaction. It is left to the object for which calls it behaves transactional. *Transactional servers* and *recoverable servers* are implemented using TOs .
- A TO which is affected by a commit or a rollback of a transaction is called a **recoverable object (RO)**.
- A **transactional server (TS)** consists of one or more objects involved in a transaction, but doesn't have any state information about the transaction.
- A **recoverable server (RS)** includes at least one RO.
- A **transactional client (TC)** can be any program which calls methods of transactional objects in the context of a single transaction.

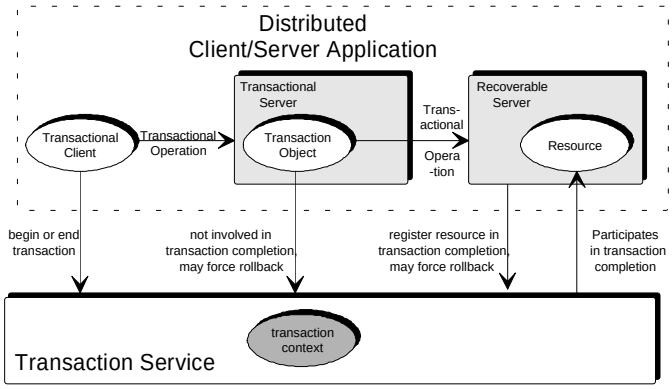

Figure 3: The Components and Objects of an OTS

To use OTS within the protocol steps of our architecture, a correct assignment of the different components is necessary. The originator host of an agent migration is the initiator of a transaction and acts as a transactional client. The target host of the migration and the trust service act as recoverable servers in the transaction. The respective recoverable objects are an agent receiver with the resource object "copy of the agent" at the target host and the account manger with the resource object "agent account" at the trust service. A realization of the recoverable servers is much easier than the implementation of a distributed database. Due to the fact that there is no multi-user access on the recoverable objects no concurrency control is necessary. Only rollback and recovery mechanisms must be available for the recoverable objects.

Figure 4 shows the transactional operations:

1. The originator host acts as a transactional client and starts the transaction. In the OTS a so-called *Thread of Control* (ToC) will be created, which is assigned to this specific transaction.
2. The transaction context is transmitted to the recoverable objects with each request.
3. The recoverable objects (*Account Manager* and *Agent Receiver*) register their resource objects (*Account of the Agent* respective *Agent*) for the agent completion. They may force a rollback of the transaction.

4. The resource objects participate in the transaction completion (2PC).

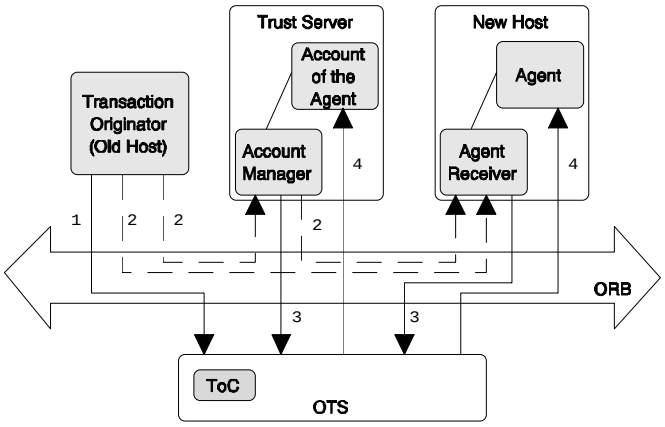

Figure 4: Objects of our system in the context of OTS

One drawback of distributed transaction processing results in a loss of performance. However, for the transport of a mobile agent there exist no real-time requirements, so this problem is less significant for our protocol. Also the additional loss of performance in relation to a normal agent transfer without transactions is not significant. Exact performance data will be available with the first benchmarks of different mobile agent systems. Nevertheless, distributed transaction processing and the 2PC protocol offer a standardized and easy-to-handle way to cope with distributed data in a consistent manner.

4.2 Scaling for large distributed systems

The design of our architecture is suitable for scaling in large distributed systems like the Internet. Similar to CORBA or OTS our trust service is responsible for a specific *agent domain*, which can cooperate with other domains. In OTS, all instances of a transaction, servers as well as clients, pertain to one domain. It is important to see that an OTS domain is absolutely independent of a CORBA domain. If several CORBA domains can interoperate, this enables to span one OTS domain over the various CORBA domains.

When an agent itinerates across an agent domain, which can be similar to a CORBA domain, the control of this agent will be given to the new trust service. This includes that the new trust service will be involved in the agent transfer and the corresponding distributed transaction. Figure 5 illustrates the relationship between an agent and an OTS domain.

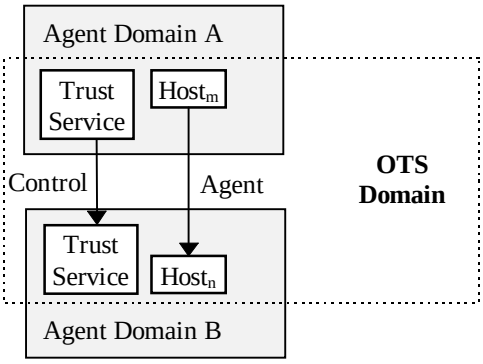

Figure 5: Agent and OTS Domain

5 Conclusion and Future Work

We presented an architecture for a mobile agent system which is highly secure against external attacks. The integrity and internal security is guaranteed by a trust service and by logging all relevant information. So in case of a fraud the offender can be ascertained and called to account. Based on this procedure our architecture offers the possibility to make money transactions for an agent, and to be a part of the electronic market place. As a further profit, high-level mechanisms for fault tolerance introduced by distributed transaction processing and different encryption methods, are provided.

As future work we see the integration of special security models for mobile agents, like a *personal security assistance* [RaJa96], which observes an agent and permits critical commands only after a check by an expert. We also want to integrate the security policies of different agent languages like Java [ArGo96] and TCL [OLW96].

In [VKM97] we already proposed an alternative realization of our system based on the recently published *Transaction Internet Protocol* (TIP) by Microsoft and Tandem [LEK97]. The *Internet Engineering Task Force* (IETF) currently examines to standardize TIP for the Internet. There exists also a public accessible reference implementation of TIP in JAVA [TIP97]. We also will investigate an alternative of our system based on DCE [RFK92] and the X/Open DTP standard [XOP96].

Further enhancements of our architecture are the possibility of a specific duplication of an agent, which can easily be applied to the protocol, and which is supported by the distributed transaction processing mechanism. Several agents now work together to fulfill the same goal, using the trust service as a location service for the different instances of a specific agent.

Literature

[ArGo96] K. Arnold, J. Gosling: *The Java Programming Language*, Addison-Wesley, ISBN 0201-63455-4, 1996

[BeNe96] P. Bernstein, E. Newcomer: *Principles of Transaction Processing*, ISBN 1-55860-415-4, Morgan Kaufmann Publisher, 1996

[BoRo93] N. S. Borenstein, M. T. Rose: *MIME Extensions for Mail-Enabled Applications: application/Safe-Tcl and multipart/enabled-mail*, Distributed as part of the Safe-Tcl 1.2, November 1993, Working Draft

[Bor94] N. S. Borenstein: *EMail With A Mind of Its Own: The Safe-Tcl Language for Enabled Mail*, ULPAA '94, Barcelona

[DEC96a] Digital Equipment Corporation*: Enterprise TP on the Internet*, http://www.software.digital.com/tpi/TPI_CO.HTMl, 1996

[DEC96b] Digital Equipment Corporation: *MILLICENT Digital's Microcommerce System* http://www.research.digital.com/SRC/millicent/, 1996

[Dig96] DigiCash Home Page: http://www.digicash.com/

[FGS96] W. Farmer, J. Guttman, V. Swarup: *Security for mobile agents: Issues and requirements*, In Proc. of the 19th National Information Systems Security Conference, Baltimore, MD, 1996

[FV96] First Virtual Home page: http://www.fv.com

[Gar94] S. Garfinkel: *PGP: Pretty Good Privacy*. ISBN 1-56592-098-8, O'Reilly & Associates, 1994

[GrRe93] J. Gray, A. Reuter: *Transaction Processing: Concepts and Techniques*, ISBN 1-55860-190-2, Morgan Kaufmann Publisher, 1993

[HCK95] C.D. Harrison, D.M. Chess, A. Kershenbaum: *Mobile Agents: Are they a good idea?*; IBM Research Report #RC 19887, IBM Research Division, 1995

[Hu95] W. Hu: *DCE Security Programming*, ISBN 1-56592-134-8, O'Reilly & Associates, 1995

[IBM96] IBM Research Hawthorne and Zürich: *Internet Keyed Payment Protocols*, http://www.zurich.ibm.com/Technology/Security/extern/ecommerce/iKP.html, 1996

[Kal93] B. Kaliski: *Privacy Enhancement for Internet Mail: Part IV: Key Certification and Related Services*, RFC 1424, RSA, February 1993

[KaWh96] R. Kalakota, A. B. Whinston: *Frontiers of Electronic Commerce*, ISBN 0-201-84520-2, Addison-Wesley Publishing Company, Inc., 1996

[Lan96] D. B. Lange: *Agent Transfer Protocol* ATP/0.1 Draft, http://www.ibm.co.jp/trl/aglets/atp/atp.html, July 1996

[LDD95] A. Lingnau, O. Drobnik, P. Dömel: *An HTTP-based Infrastructure for Mobile Agents*, World Wide Web Journal - 4th Int. World Wide Web Conference Proceedings, Boston, December 1995

[LEK97] J. Lyon, K. Evans, J. Klein: *Transaction Internet Protocol*, Internet-Draft, http://204.203.124.10/pdc/docs/TIP.txt

[MaHa96] P. Mandl, B. Hackler: *Transaktionsorientierte Managementsysteme und Managementprotokolle*, Praxis der Informationsverarbeitung und Kommunikation, Vol. 19, No. 4, 1996

[MüSc92] M. Mühlhäuser, A. Schill*: Software Engineering für verteilte Anwendungen*, ISBN 3-540-55412-2, Springer Verlag, 1992

[NeTs94] B. C. Neuman and T. Ts'o, Kerberos: *An Authentication Service for Computer Networks*, IEEE Communications Magazine, Volume 32, Number 9, 1994

[OLW96] J. K. Ousterhout, J. Y. Levy, B. B. Welch*: The Safe-Tcl Security Model*, http://www.sunlabs.com/research/tcl/SafeTcl.ps, Draft, November 1996

[OMG94] Object Management Group: *Object Transaction Service*, 1994

[OMG95] Object Management Group: *The Object Request Broker : Architecture and Specification*, Revision 2.0, 1995

[PeSt97] H. Peine and T. Stolpmann: *The Architecture of the Ara Platform for Mobile Agents*, Proc. of the First Int'l. Workshop on Mobile Agents MA'97 Berlin, LNCS No. 1219, Springer Verlag, 1997

[Pyth94] Safe-Python Homepage: http://minsky.med.virginia.edu/sdm7g/Projects/Python/SafePython.html, 1994

[RaJa96] A. Rasmusson, S. Janson. *Personal security assistance for secure Internet commerce*. In New Security Paradigms '96, ACM Press, Sept. 1996

[RKF92] W. Rosenberry, D. Kenney, G. Fisher: *Understanding DCE*, ISBN 1-56592-005-8, O'Reilly & Associates, 1992

[Riv92] R. Rivest: *The MD5 Message-Digest Algorithm*, RFC 1321, MIT, April 1992

[SBH96] M. Straßer, J. Baumann, F. Hohl: *Mole - A Java based Mobile Agent System*, Proc. of the ECOOP '96 Workshop on Mobile Object Systems, 1996

[Sch96] B. Schneier: *Applied Cryptography*, ISBN 0-471-11709-9, J. Wiley & Sons, Inc., 1996

[TIP97] Transaction Internet Protocol, Reference Implementation, http://204.203.124.10/pdc/docs/TIP.zip

[VKM97] H. Vogler, T. Kunkelmann, M.L. Moschgath, *Distributed Transaction Processing as a Reliability Concept for Mobile Agents*, Proc. 6th IEEE Workshop on Future Trends of Distributed Computing Systems, Tunis, Oct. 1997

[Vog96] F. Vogt: *Werkzeuge für die Transaktionsverarbeitung heute - morgen*; Proc. of the 19th European Congress Fair of Technical Communication - ONLINE'96, Congress VI, Hamburg, Feb. 1996

[Whi94] J. E. White: Telescript Technology: *The foundation for the electronic Marketplace*, White Paper. General Magic Inc., 1994

[XOP96] X/Open Guide: *Distributed Transaction Processing: Reference Model*, Version 3, X/Open Company Ltd., 1996