BAMBERGER BEITRÄGE

ZUR WIRTSCHAFTSINFORMATIK UND ANGEWANDTEN INFORMATIK

ISSN 0937-3349

Nr. 66

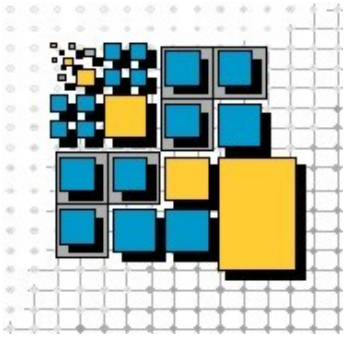
Concealing Presence Information in Instant Messaging Systems

Markus Dorsch, Martin Grote,
Knut Hildebrandt, Maximilian Röglinger,
Matthias Sehr, Christian Wilms,
Karsten Loesing, and Guido Wirtz

April 2006

FAKULTÄT WIRTSCHAFTSINFORMATIK UND ANGEWANDTE INFORMATIK

OTTO-FRIEDRICH-UNIVERSITÄT BAMBERG



Distributed and Mobile Systems Group

Otto-Friedrich Universität Bamberg

Feldkirchenstr. 21, 96052 Bamberg, GERMANY

Prof. Dr. rer. nat. Guido Wirtz

http://www.uni-bamberg.de/en/fakultaeten/wiai/faecher/informatik/lspi/

Due to hardware developments, strong application needs and the overwhelming influence of the net in almost all areas, distributed and mobile systems, especially software systems, have become one of the most important topics for nowadays software industry. Unfortunately, distribution adds its share to the problems of developing complex software systems. Heterogeneity in both, hardware and software, concurrency, distribution of components and the need for interoperability between different systems complicate matters. Moreover, new technical aspects like resource management, load balancing and deadlock handling put an additional burden onto the developer. Although subject to permanent changes, distributed systems have high requirements w.r.t. dependability, robustness and performance.

The long-term common goal of our research efforts is the development, implementation and evaluation of methods helpful for the development of robust and easy-to-use software for complex systems in general while putting a focus on the problems and issues regarding the software development for distributed as well as mobile systems on all levels. Our current research activities are focussed on different aspects centered around that theme:

- *Robust and adaptive Service-oriented Architectures*: Development of design methods, languages and middleware to ease the development of SOAs with an emphasis on provable correct systems that allow for early design-evaluation due to rigorous development methods and tools. Additionally, we work on approaches to autonomic components and container-support for such components in order to ensure robustness also at runtime.
- *Agent and Multi-Agent (MAS) Technology:* Development of new approaches to use Multi-Agent-Systems and negotiation techniques, for designing, organizing and optimizing complex distributed systems, esp. service-based architectures.
- *Peer-to-Peer Systems*: Development of algorithms, techniques and middleware suitable for building applications based on unstructured as well as structured P2P systems. A specific focus is put on privacy as well as anonymity issues.
- *Context-Models and Context-Support for small mobile devices*: Investigation of techniques for providing, representing and exchanging context information in networks of small mobile devices like, e.g. PDAs or smart phones. The focus is on the development of a truly distributed context model taking care of information reliability as well as privacy issues.
- *Visual Programming- and Design-Languages*: The goal of this long-term effort is the utilitization of visual metaphores and languages as well as visualization techniques to make design- and programming languages more understandable and, hence, easy-to-use.

More information about our work, i.e., projects, papers and software, is available at our homepage. If you have any questions or suggestions regarding this report or our work in general, don't hesitate to contact me at guido.wirtz@wiai.uni-bamberg.de

Guido Wirtz

Bamberg, April 2006

Concealing Presence Information in Instant Messaging Systems

Markus Dorsch, Martin Grote,
Knut Hildebrandt, Maximilian Röglinger,
Matthias Sehr, Christian Wilms,
Karsten Loesing, and Guido Wirtz

Lehrstuhl für Praktische Informatik, Fakultät WIAI

http://www.lspi.wiai.uni-bamberg.de/dmsg/software/hashky/

Abstract Information about online presence allows participants of instant messaging (IM) systems to determine whether their prospective communication partners will be able to answer their requests in a timely manner, or not. That is why presence information, combined with the ability to send instant messages, makes IM more personal and closer than other forms of communication such as e-mail. On the other hand, revelation of presence constitutes a potential of misuse by untrustworthy entities. A possible threat is the generation of online logs disclosing user habits. This makes presence information a resource worth protecting. We argue, however, that current IM systems do not take reasonable precautions to protect presence information.

We implemented an IM system designed to be robust against attacks to disclose a user's presence. In contrast to existing systems, it stores presence information in a registry in a way that is only detectable and applicable for intended users and not comprehensible even for the registry itself. We use a distributed hash table (DHT) as registry and apply an anonymous communication network to protect the physical addresses of both senders and receivers of messages.

Keywords Instant Messaging, Presence Information, Anonymity, Peer-to-Peer

Contents

1 Introduction

Presently, instant messaging (IM) services are evolving from an informal chat tool to a widely accepted communication medium. According to several surveys, the employment of IM for business communication is steadily increasing and has the potential to replace email as currently mostly employed communication medium. In this context, security problems become more obvious and critical.

One important component of IM services is presence information of potential communication partners which constitutes a non-negligible additional value compared to other synchronous communication media, e.g. the telephone. In case of telephones, the availability of the interlocutor cannot be determined until the actual communication takes place. It is for this reason that other services like Voice over IP (VoIP) (see Skype[1]) integrate presence information more and more.

This fact makes presence information a resource worth securing in IM services. The creation of online logs of certain users could be considered as an imaginable misuse that allows the deduction of private or non-private behavioral patterns. Current IM and VoIP services, like ICQ[2], Yahoo[3], MSN[4], and Skype, deposit their users' presence information on physically central entities and barely make arrangements to prevent misuse.

This project aims at realizing confidentiality of presence information on the one hand as well as private communication between communication partners on the other hand. This technical report describes the technical realization rather than the design decisions of the project. For a more elaborate description of the underlying concepts refer to [LDG+06].

The remainder of this paper is structured as follows: Section 2 summarizes and elaborates the requirements of this project, whereas section 3 discusses the most important design decisions. Afterwards, section 4 introduces the architecture of the system. Section 5 describes client-specific issues and preparatory activities. Sections 6 and 7 give a detailed survey of the required protocols between the involved parties. Section 8 describes possible attacks on the system. Finally, section 9 concludes the report.

2 Requirements

From a functional point of view, our IM service has to implement the basic functions provided by existing IM services. In particular, this includes the *sharing of presence information* as well as the *exchange of instant messages* with other users.

Confidentiality of presence information as well as private communication between communication partners are further requirements.

[1] `http://www.skype.com`
[2] `http://www.icq.com`
[3] `http://messenger.yahoo.com/`
[4] `http://messenger.msn.com`

More precisely, all users of the IM service must be provided with a relatively high degree of *anonymity* [PH05] against attackers from inside and outside the system. The presence status of a user must be neither directly visible nor indirectly observable or deducible for attackers. This requires the camouflaging of presence information on a logical and a physical system layer. Further, concerning instant message exchange, both sender and receiver of a message must remain anonymous during communication. Their physical locations[5] must be neither revealed nor directly accessible to both parties.

Additionally, the IM service must guarantee its users' anonymity with a certain degree of robustness. In particular, a central entity that is responsible for storing presence or contact information and therefore constitutes a single point of failure is not sufficient. The higher the level of robustness required, the more administrative overhead the IM service will have to manage. Hence, there is a trade-off between system efficiency and level of robustness. As a last aspect, anonymity should be as transparent to the user as possible and barely cause additional effort.

To summarize, the basic non-functional goal of the IM service is to provide its users a high level of anonymity by employing techniques of encryption and camouflaging on both, logical and physical system layers.

3 Design Decisions

The requirements described in section 2 have an impact on the design of the IM service. This section provides a survey of the most important design decisions. After that we describe the structure of the hash keys and values stored in the Chord network.

3.1 Essential Design Decisions

As described in the requirements, a centralized client-server approach does not fulfill the requirements of a flexible, robust, and attacker-resistant system architecture, because it constitutes a single point of failure. Hence, the IM service is based on *peer-to-peer* (P2P) technology. As a consequence, the clients exchange instant messages and share their presence statuses only directly and not intermediated by a centralized server. Though being less efficient, this solution supports the P2P approach, in particular robustness.

A *distributed hash table (DHT)* is employed for the storage and retrieval of information in the P2P system because it provides a comfortable and easy-to-use interface and places few constraints on the structure of keys and data to be stored. As an efficient DHT a *Chord* [SMLN+03] implementation is available for the project. Client nodes are not part of the Chord network, because they should be connected to as few other nodes as possible.

In order to fulfill the anonymity requirements, *Tor* [DMS04] was chosen as anonymous com-

[5] Physical location means a physical address, e.g. IP-address and port

munication network which is based on onion routing. It realizes sender *and* receiver anonymity unlike other anonymization tools like Java Anon Proxy (JAP)[6] which only provides anonymity of the sender.

3.2 Structure of a Hash Key

A hash key consists of the following components:

- The *publisher ID* is a unique identifier for the publisher of contact information.
- The *receiver ID* is a unique identifier for the receiver of contact information.
- The *common secret* is only known to the publisher and the receiver of the contact information. It is needed to enable only the publisher and the receiver to generate the hash key where the contact information is stored. The publisher and the receiver have to agree on the common secret before the initial communication. Moreover, it should be changed from time to time.
- The *dynamic component* is an integer value that causes the hash key to change during time and hence modify its position in the DHT. The dynamic component may be public (to avoid even more agreement overhead). It is worthless for an intruder that does not know the common secret.

  Without the dynamic component the contact information for a specific contact would always be stored on the same node in the Chord network. From this, an evil Chord node or an evil observer outside the system would gain presence information and therefore violate the anonymity goal of the IM service. Hence, it is inevitable that the client's contact information "changes its position" in the Chord network. The only way to do this is to periodically (but systematically) change the hash key. Therefore the structure of the hash key includes a dynamic component.

  A straightforward possibility to implement the dynamic component is for example to take the hour component of the global time, because it increases monotonically and needs no further agreement of the involved parties.
- The *index* is a set of predefined integer values (e.g. 0–4) that are used to store the hash value on different nodes in the Chord network. The index must not be confounded with the dynamic component. The index realizes some way of replication so that no single node of the Chord network (which of course is possibly evil) is able to compromise a client's contact information.

In order to conceal the values of the described components the plain key has to be hashed by a hash function, e.g. SHA-1.

[6] `http://anon.inf.tu-dresden.de/`

3.3 Structure of a Hash value

A hash value consists of the following components:

- The *timestamp* allows the receiver of the contact information to verify which entry in the DHT is the most recent. This is necessary, because an evil Chord node may discard entries or at least hold them back for some time.
- The *contact information* allows its receiver to establish communication with the publisher of the contact information. Currently, the contact information is a Tor-specific URL.
- The *index* is a set of integer values that are used to modify the encrypted and signed hash value. Otherwise identical hash values are associated to different hash keys (because of the hash key's indices). An attacker that controls several Chord nodes could retrieve several contact information entries (because they have identical hash values) and therefore compromise the contact information.

The hash value is signed with the publisher's private key and encrypted with the receiver's public key in order to ensure that only the designated client is able to decrypt the hash value and to avoid that an evil Chord node modifies the contact information maliciously.

A combination of *symmetric and asymmetric encryption* and signing technologies was chosen to ensure secure message exchange between communication partners.

4 System architecture

This section provides a global view of the system architecture of the IM service and is structured as follows: Firstly we introduce the basic roles, explain their functions, and deduce the corresponding protocols that organize the interplay between the roles. Next, we describe the interplay of these roles.

4.1 Roles of the IM service

The architecture of the IM service consists of three roles which have their own characteristics and responsibilities. Each role has to perform specific functions to contribute to the overall operability of the system. The three roles are:

- Clients
- Chord nodes
- Tor nodes

Further, the protocol relies on the following servers being present in the network, possibly replicated on multiple nodes:

- Chord node list server
- Time server
- PGP server

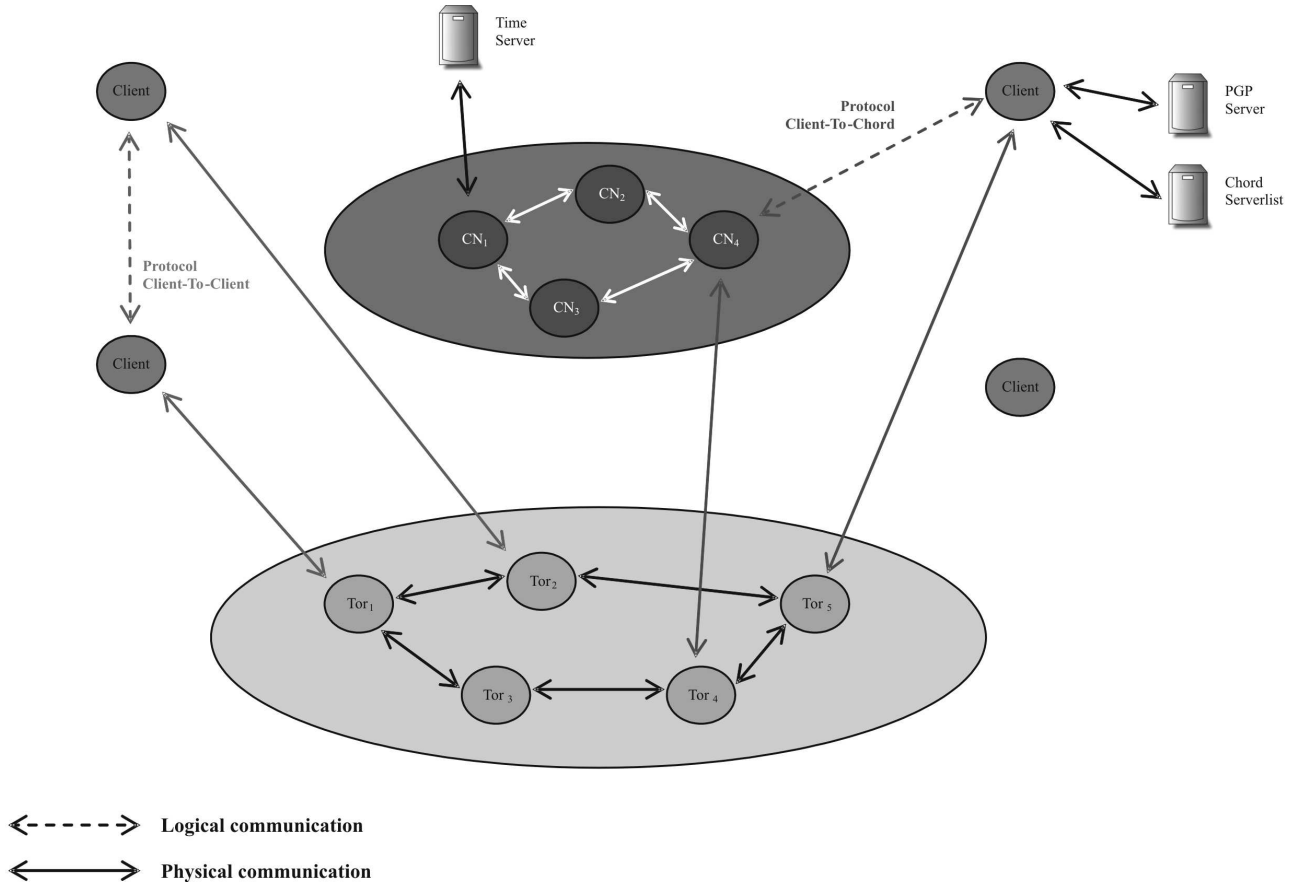


Figure 1: System architecture

Figure 1 shows the system architecture including the roles. In order to organize the interplay between the roles three different protocols have to be formalized (the *Client-to-Chord protocol*, the *Client-to-Client protocol*, and the Chord protocol itself). Referring to the design decisions and the requirements, the Tor network is assumed to be given. Hence, the Tor protocols are transparent for the IM service.

Client: A client represents a user in the system who shares his presence information and exchanges instant messages with other users. Therefore, a client keeps a local list of all of the user's contacts (referred to as contact list in the rest of the document).

The client[7] is responsible both for sharing its user's presence status with all other users on the local contact list and for sending and receiving instant messages with other users. Clients

[7]In the following, the expressions client and user are used synonymously.

which communicate with other clients must conform to the Client-to-Client protocol. Moreover, a client interacts with Chord nodes (which act here as access points to the DHT) in order to initially retrieve the contact information of its contacts or to update its own contact information for offline contacts. Clients which communicate with Chord nodes must conform to the Client-to-Chord protocol. A client distinguishes several availability statuses: Available (A), Be right back (BRB), Do not disturb (DND), Extended Away (XA), Not Available (NA), and Offline (OFF). Note that the communication between clients and Chord nodes is always realized using anonymous communication.

Chord node: Referring to the design decisions, a DHT is employed to store and retrieve contact information. The DHT is implemented and organized by a set of Chord nodes. Each node is locally responsible for a set of entries with affiliated IDs. Further, each node knows its neighbors and other nodes in order to build up the ring structure and to enable the search for data.

Tor node: As described in the requirements, confidentiality of presence information and private communication must be guaranteed by providing sender and receiver anonymity. To implement anonymity, we choose the external Tor network project which is based on the onion routing concept[8]. Here, instead of taking the direct way from the sender to the receiver, messages follow a randomly chosen path through the Tor network. A message can be considered as a nested object where each layer encapsulates the following layers and the next destination within the path. Upon receiving a message, a Tor node (also known as onion router) removes the outmost layer—which is the only layer that can be removed with the Tor node's private key—and forwards the message to the next router or the final receiver. Hence, a Tor node knows neither its position in the path nor the complete path. This makes it difficult for observers to determine identity and location of the sender and the receiver.

4.2 Interplay of the roles

Figure 1 illustrates the interplay of the described roles (note that users have to agree upon a common secret outside the IM service before the interaction within the IM service is possible). Clients logically communicate with other clients to share presence information and to exchange instant messages. The physical communication is based on the Tor network to guarantee anonymity. Further, clients logically communicate with Chord nodes to store and retrieve contact information in and from the DHT. Finally, clients communicate with key servers and Chord node list servers in order to retrieve the keys of yet unknown contacts and the list of available Chord nodes. Chord nodes communicate to organize the DHT and to request the global time of external time servers. As Chord nodes are known to each other, no anonymization over the Tor network is needed.

[8]For detailed information see `http://www.onion-router.net/`

5 Standard Use Cases

Clients which want to take part in the IM service have to perform several preparatory activities. Additionally, the behavior of a client has to conform to a sequence of actions during a single communication session.

5.1 Preparatory Activities

In order to communicate and to exchange presence information, two clients have to perform the following steps: At first, they have to exchange their identifiers because there is neither a central nor a distributed lookup service for user identifiers implemented in the IM service. After that, they have to agree upon a common secret which is exclusively known to them and therefore should be changed in regular intervals. Finally, the clients may have to exchange their public keys, if necessary. In order to keep the system as simple as possible, it is assumed that the client accomplishes all of these activities outside the IM service, e.g. over email.

5.2 Single Communication Session

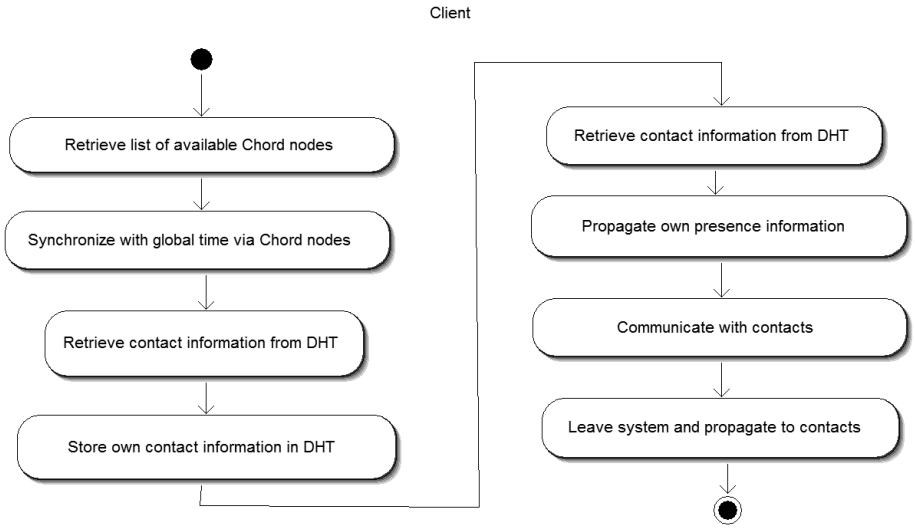


Figure 2: Client behavior during a single communication session

Figure 2 illustrates the sequence of a standard communication session of a client.

Retrieve list of available Chord nodes: Initially, before entering the IM service, the client has to retrieve a list of the available Chord nodes from a previously known server.

Synchronize with global time via Chord nodes: The client has to synchronize its local time with the global time. This involves an indirect communication with external time servers via known Chord nodes, because a direct communication would reveal the presence of the client. This is part of Client-to-Chord protocol.

Retrieve contact information from DHT: The client retrieves the contact information of all of its contacts from the DHT. Using this information the client concludes how to reach each of its contacts in order to notify them of its presence status and to retrieve their presence statuses. This is part of Client-to-Chord protocol.

Store own contact information in DHT: If there is no contact information for a contact in the DHT, he is assumed to be offline, i.e. at least there is no possibility for the client to establish a communication. The client stores its own contact information for each offline contact in the DHT. Another client that logs into the system at a later time is able to establish a communication with this contact information. This is part of Client-to-Chord protocol.

Retrieve contact information from DHT (Second time): In order to prevent the situation that two clients miss themselves out, it is necessary that a client which has left its contact information in the DHT, retrieves the contact information of contacts considered offline a second time. This is part of Client-to-Chord protocol.

Propagate own presence information: If there are entries in the DHT, the contact is possibly online in the IM service. More precisely, the contact is not necessarily online, because the contact information may be out of date. This may occur in two cases: Firstly, if the contact crashes. Secondly, if the contact has successfully signed off and the garbage collection of the DHT has not yet deleted the obsolete entries. The client informs each contact who is possibly online of his presence status. While being present in the IM service, a client periodically[9] updates its contact information in the DHT for each offline contact. Note that the client itself never deletes any entry from the DHT, because this would require a means for authorization which we wanted to avoid. This task is undertaken by the DHT's garbage collection. This is part of Client-to-Client protocol.

Communicate with contacts: The client sends and receives instant messages to and from other contacts. This is part of Client-to-Client protocol.

Leave the system and propagate to contacts: The client has to inform each online contact that it leaves the system. There is no further interaction with the Chord nodes, because the obsolete entries with its contact information will be deleted by the DHT's garbage collection. This is part of Client-to-Client protocol.

6 Client-to-Chord Protocol

This section describes protocol sequences that occur in the Client-to-Chord protocol. Table 1 contains the messages which are exchanged in this protocol.

[9]The interval of the DHT's garbage collection could be harmonised with the dynamic component in order to reduce update traffic

RPC Type	Purpose	Initiator	Responder	Request parameters	Reply parameters
RETRIEVE	Requests the retrieval of an entry by its unique key	Client	Chord node	Hash key of the hash values to be retrieved	Hash values corresponding to hash key; Status
INSERT	Requests the insertion of a new entry	Client	Chord node	Hash key of the entry to be inserted; Hash value of the entry to be inserted	Completion state (successful or failed); Cause of a failed request
TIME SYNCHRONIZATION	Requests current time to synchronize local time	Client	Chord node	—	Current time

Table 1: Client-to-Chord Message Types

6.1 Protocol Sequence: Retrieve Contact Information

When logging into the system, a client retrieves the contact information of its contacts from the DHT. Using this information the client concludes how to reach the contact in order to notify it of its presence and request its contacts' presence.

In order to ensure resilience against evil Chord nodes, the client has to check several entries in the hash table for each contact, because an evil Chord node could deliberately discard previously stored entries or at least hold them back for some time. Different hash keys are generated for one contact information by alternating or incrementing the index value.

The client repeats the following steps for each contact:

1. Generate the hash keys for the request. As explained above, several hash keys have to be generated by altering the index value.
2. Create as much RETRIEVE REQUEST messages as there are hash keys and encapsulate each generated hash key in one message. Then, send all messages to different known Chord nodes. It is important to choose different Chord nodes, because a single evil Chord node receiving all messages could simply discard them and hence reject the entire request.

After receiving an incoming RETRIEVE REQUEST message a Chord node has to perform the following steps:

3. Retrieve the hash value corresponding to the requested hash key. This is part of the Chord-to-Chord protocol.
4. Create a RETRIEVE REPLY depending on the result of the retrieval:
   (a) If a hash value corresponding to the requested hash value has been found, create a RETRIEVE REPLY message that encapsulates the hash value and set its state to SUCCESSFUL.

   (b) If no hash value has been found, create a RETRIEVE REPLY message and set its state to FAILED.

5. Send the RETRIEVE REPLY message to the client

The client processes the received RETRIEVE REPLY message:

6. If the state is SUCCESSFUL, the contact is considered to be possibly online. More precisely, the contact is not necessarily online, because the contact information may be out of date. This may occur in two cases: Firstly, if after the contact's crash the garbage collection of the DHT has not yet deleted the obsolete entries. Secondly, if after the contact's successful sign off the garbage collection of the DHT has not yet deleted the obsolete entries. The client has to inform the contact of his own presence status.

7. If the state is FAILED, the contact is considered to be offline. To ensure that the contact is able to establish communication with the client when it goes online, the client has to store its own contact information in the DHT. The detailed protocol sequence is explained next.

6.2 Protocol Sequence: Publish Own Contact Information

For each contact that is considered to be offline the client has to store its own contact information in the DHT, so that the contact can establish communication with the client when it logs into the system by itself. In order to ensure robustness against evil Chord nodes, the client has to store several entries in the DHT for each offline contact, because an evil Chord node could deliberately modify or discard stored entries or at least hold them back for some time. Different hash keys for a contact information are generated by incrementing the index value.

The client repeats the following steps for each contact:

1. Generate the hash keys for the request. Several hash keys are generated by altering the index value.

2. Generate the hash value. After that, the hash value is signed with the sender's private key and encrypted with the contact's public key.

3. Create as much INSERT REQUEST messages as there are hash keys and encapsulate each generated hash key together with the hash value in one message. Send all messages to different known Chord nodes.

After receiving an incoming INSERT REQUEST message a Chord node has to perform the following steps:

5. Insert the hash value and the hash key in the DHT.

6. Create an INSERT REPLY depending on the insertion state:
   (a) If the insertion was successful, create an INSERT REPLY message and set its state to SUCCESSFUL.
   (b) If the insertion was not possible for a Chord-specific reason, create an INSERT REPLY message and set its state to FAILED.
7. Send the INSERT REPLY message to the client

Finally, the client processes the received INSERT REPLY message depending on its state:

8. If the state is SUCCESSFUL, there is nothing more to do.
9. If the state is FAILED, the client contacts another Chord node.

During presence in the IM service, a client periodically updates its contact information in the DHT. For each offline contact it stores again the specified number of entries.

6.3 Protocol Sequence: Time Synchronization

A client does not request the time service directly in order to perform time synchronization but indirectly via a Chord node.

In order to retrieve the current time the client has to perform the following steps:

1. The client creates a request for time synchronization to a Chord node.
2. The Chord node requests the current time from the time service and replies to the client request.

7 Client-to-Client Protocol

Clients communicate with other clients to share presence information and to exchange instant messages. The Client-to-Client protocol specifies the message transfer between clients. Table 2 contains the messages which are exchanged in this protocol.

7.1 Protocol Sequence: Generate Session Key

Before a client is able to establish a communication with one of its contacts, it has to exchange a symmetric session key with it. The session key is valid until one of the two communication partners goes offline (or crashes).

RPC Type	Purpose	Initiator	Responder	Request parameters	Reply parameters
INIT SESSION	Initializes a session with the receiver by exchanging the symmetric session key	Client which aims at opening the session	Client with which the session shall be opened	Symmetric session key (encrypted with public key), Initial status	Acknowledgment for a session key (encrypted with symmetric session key)
NOTIFY STATUS	Notifies the receiver of the current presence status of the sender	Client which changes or renews its presence status	Client which is to be informed of the presence status change or renewal	Presence status (encrypted with symmetric session key)	—
INSTANT MESSAGE	Conveys an instant message from the sender to the receiver	Client which sends an instant message	Client which receives an instant message	Instant message text (encrypted with symmetric session key)	—

Table 2: Client-to-Client Message Types

In order to exchange a session key a client performs the following tasks:

1. Generate a symmetric session key.
2. Create an INIT SESSION message containing the session key and a timestamp to prevent replay attacks.
3. Sign the message with the private key of the client and encrypt it with the contact's public key.
4. Send the message to the contact.

Upon receiving an INIT SESSION message, the contact performs the following tasks:

5. Decrypt the message with the contact's private key and verify the signature with the public key of the sending client.
6. Check if the timestamp is up to date. If it is out of date, ignore the message.
7. Create an INIT SESSION REPLY message.
8. Encrypt the message with the received symmetric session key.

9. Send the message to the client.

Upon receiving the INIT SESSION REPLY message, both the clients use the symmetric session key instead of the contact's public key for the rest of the session.

7.2 Protocol Sequence: Notify Of Presence Status

A client is responsible for periodically propagating its own presence status to all other clients which it considers to be "online" when it initially goes "online" as well as during presence in the IM service. For each contact which is considered to be "online" the client repeats the following steps:

1. Create a NOTIFY STATUS message containing the current presence status of the client and a timestamp to prevent replay attacks.
2. Encrypt the message with the previously generated session key.
3. Send the message to the receiver.
4. The further proceeding depends on the state of the message delivery.
   (a) If the message delivery was successful, the client waits a constant interval before renewing its presence information. Afterwards, the client checks in its local contact table whether the contact is still considered to be "online".
      i. If the contact is still considered to be "online", the client starts over renewing its own presence status.
      ii. If the contact is now considered to be "offline", i.e. it has apparently signed off, the client can stop renewing its presence status for this contact and has to leave its contact information in the DHT.
   (b) If the message delivery failed, i.e. the contact has apparently crashed, the client has to mark the contact as "offline" in its local contact list and leave its own contact information for the contact in the DHT.

Upon receiving the NOTIFY STATUS message the contact has to do the following steps:

1. Decrypt the message.
2. Check if the timestamp is higher than the timestamp of the last received message from the corresponding contact. If it is not ignore the message.
3. Update local contact list.

7.3 Protocol Sequence: Send Instant Message

If a client wants to send an instant message to another client, it performs the following steps:

1. At first, the client must get the presence status of the contact to which it wants to send a message from its local contact list.
   (a) If the contact is ”offline”, the message will not be sent.
   (b) If the contact is ”online”, the client creates an INSTANT MESSAGE that encapsulates the instant message text and a timestamp to prevent replay attacks.
      i. The client encrypts the message with the previously generated session key.
      ii. The further proceeding depends on the status of the message delivery.
         A. If the message delivery was successful, the protocol sequence is completed.
         B. If the message delivery has failed, i.e. the contact has apparently crashed, the client marks the contact as ”offline” in its local contact list and leave its own contact information for the contact in the DHT.

Upon receiving the INSTANT MESSAGE the contact has to perform the following steps:

1. Decrypt the message.
2. Check if the timestamp is higher than the timestamp of the last received message from the corresponding contact. If it is not ignore the message.

7.4 Protocol Sequence: Going Offline

If a client goes ”offline”, it notifies all of its ”online” contacts directly. In the Client-to-Client protocol there is no extra message type for an ”offline” notification. Instead, a client sends a NOTIFY STATUS message containing ”offline” as presence status.

8 Possible Threats

As described in the introduction, both confidentiality of presence information and private communication between communication partners are the nonfunctional goals of our IM service. It is important to examine the possible attacks on the IM service and analyze which it can withstand and up to which degree.

8.1 Attacks From Inside the IM Service

This section describes attacks from inside the IM service. That is, an attacker maliciously impersonates a client, one or more Chord nodes, or one or more Tor nodes and tries to directly gain information by confusing the other parties. Since, in this context an attacker impersonates one of the system's roles, all attacks are performed on the logical system layer.

Infiltration of a client: If an attacker impersonates an existing client, it may strive for two different goals. On the one hand, it may want to retrieve the contact information and the presence status of the client's contacts. Therefore, the attacker would have to get access to the entries which are stored in the DHT and decrypt them properly. On the other hand, the attacker might not be interested in gaining information, but just in confusing other clients so that they are not able to work properly any more.

In general, we assume that the attacker neither knows the common secret nor the private key. The client's unique identifier and the receiver's unique identifier are not secure and hence accessible to the attacker. But as long as the attacker does not know the common secret of the client, it will not be able to generate hash keys and retrieve the corresponding entries from the DHT. Even if the attacker successfully generated the correct hash keys by chance, it would not be able to decrypt the retrieved hash values properly.

The attacker might know the common secret of the client and one of its clients. As long as the private key remains secret, the attacker will not be able to sign the hash values properly. Moreover, the attacker is not able to decrypt entries from the DHT designated to the client. But presence information from the client's contacts is indirectly deducible from the fact that there are entries in the DHT.

In the worst case, the attacker knows the private key and the common secret. The attacker is now able to retrieve the corresponding entries from the DHT and decrypt the contact information. But still no physical address can be extracted because an onion address is used. Furthermore, the attacker is able to place wrong contact information in the DHT and sign it properly.

Infiltration of the Chord network: An attacker could also impersonate one or more nodes of the Chord network. In this situation the attacker may have two different goals: On the one hand, he may want to gain information about originators and receivers of contact information. On the other hand, the attacker may want to intercept, discard, modify, or delay information to be stored or retrieved in or from the DHT, which is crucial for the IM service.

First, we assume that the attacker is interested in information about originators and receivers of entries. It is impossible for the attacker to succeed because the Tor network guarantees sender and receiver anonymity, i.e. a Tor node never knows on behalf of whom it stores or retrieves entries in and from the DHT. Furthermore, the DHT entries do not contain physical contact information, but only the onion address of the Tor network. Thus even if the attacker gets access to the common secret of a pair of clients, it will not be able to extract physical contact information.

Second, an attacker could attempt to discard, modify, or delay insertion and retrieval requests of

clients. The IM service takes arrangements to guarantee a certain degree of robustness against such attacks. A client that inserts contact information in the DHT has to store several entries with different hash keys. These hash keys are generated by alternating the index value. The fact that the information is physically stored on different nodes of the DHT assures that the information can only be discarded, modified, or delayed completely by a cooperation of evil Chord nodes This is certainly more difficult than infiltrating a single Chord node. Attacks on the retrieval of contact information have to be performed analogously.

8.2 Attacks From Outside the IM Service

As shown above, attacks from inside the IM service are not easy to perform. The other possibilty is to attack the system from outside, e.g. by eavesdropping traffic analysis. Since, in this context an attacker does not impersonate one of the roles, all attacks are performed on the physical system layer.

Eavesdropping: An attacker has several points for eavesdropping. More precisely, it can eavesdrop on the communication between clients, between clients and Chord nodes between Chord nodes. Physical communication between two clients and between a client and a Chord node always involves the Tor anonymization network. Hence, an attacker is not able to extract information of the originator or the receiver of an intercepted message packet. Encryption prevents the attacker from gaining information from message contents.

Concerning the communication between Chord nodes, eavesdropping is nearly worthless for an attacker. That is, the communication never contains plain information, but only the hash key and the signed and encrypted hash value. Anymore, no information about presence of a certain user is deducible, because the Chord nodes never know about the clients of which they are currently processing requests.

Additionally, an attacker could intercept the communication between the nodes of the anonymization network. This again will not reveal information to the attacker. First, the route through the Tor network is chosen at random and, hence, the attacker does not know where exactly to eavesdrop (in particular under the assumption that the interception of the whole traffic is not feasible for most of the attackers). Second, encryption is used between Tor nodes, i.e. an attacker will not get information from intercepted data.

Finally, the IM service must be secure against replay attacks. By replaying messages an attacker could make a client obtain wrong presence states of its contacts. Thus, all messages between clients include timestamps that have to be checked by receiving client nodes. By replaying messages intercepted between clients and Chord nodes, an attacker could store wrong contact information in the DHT. This risk is minimized by employing the dynamic component of the hash key. After the dynamic component changes, the attacker can only store obsolete entries that will not be requested any more by clients.

Traffic analysis: Traffic analysis is not easy no perform on the IM service. This is due to the permanent traffic of the Tor network in which the IM service's communication vanishes completely. Though, end-to-end traffic analyses are not impossible, i.e. an attacker observing

two interacting clients will gain information about their communication.

9 Conclusion

We implemented an IM system that aims to overcome the shortcomings of existing IM systems which do not cope with attacks to disclose a user's presence. In our system presence information is stored in a distributed registry in a way so that only intended users, not even the registry itself, can detect or use it. The employed onion routing network ensures that physical adresses of both, senders and receivers of messages, are never revealed. In summary a concept coping with previously neglected security issues has been introduced and its feasibility been proven by a running implementation.

References

[DMS04] Dingledine, R., Mathewson, N., und Syverson, P.: Tor: The Second-Generation Onion Router. In: *Proceedings of the 13th USENIX Security Symposium.* S. 303–320. 2004.

[LDG+06] Loesing, K., Dorsch, M., Grote, M., Hildebrandt, K., Röglinger, M., Sehr, M., Wilms, C., und Wirtz, G.: Privacy-aware Presence Management in Instant Messaging Systems. In: *IEEE International Parallel and Distributed Processing Symposium.* 2006.

[PH05] Pfitzmann, A. und Hansen, M.: Anonymity, Unlinkability, Unobservability, Pseudonymity, and Identity Management—A Consolidated Proposal for Terminology. August 2005. http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.23.pdf.

[SMLN+03] Stoica, I., Morris, R., Liben-Nowell, D., Karger, D. R., Kaashoek, M. F., Dabek, F., und Balakrishnan, H.: Chord: A scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. Netw.* 11(1):17–32. 2003.

A List of previous University of Bamberg reports

Bamberger Beiträge zur Wirtschaftsinformatik

Stand April 3, 2006

Nr. 1 (1989) Augsburger W., Bartmann D., Sinz E.J.: Das Bamberger Modell: Der Diplom-Studiengang Wirtschaftsinformatik an der Universität Bamberg (Nachdruck Dez. 1990)

Nr. 2 (1990) Esswein W.: Definition, Implementierung und Einsatz einer kompatiblen Datenbankschnittstelle für PROLOG

Nr. 3 (1990) Augsburger W., Rieder H., Schwab J.: Endbenutzerorientierte Informationsgewinnung aus numerischen Daten am Beispiel von Unternehmenskennzahlen

Nr. 4 (1990) Ferstl O.K., Sinz E.J.: Objektmodellierung betrieblicher Informationsmodelle im Semantischen Objektmodell (SOM) (Nachdruck Nov. 1990)

Nr. 5 (1990) Ferstl O.K., Sinz E.J.: Ein Vorgehensmodell zur Objektmodellierung betrieblicher Informationssysteme im Semantischen Objektmodell (SOM)

Nr. 6 (1991) Augsburger W., Rieder H., Schwab J.: Systemtheoretische Repräsentation von Strukturen und Bewertungsfunktionen über zeitabhängigen betrieblichen numerischen Daten

Nr. 7 (1991) Augsburger W., Rieder H., Schwab J.: Wissensbasiertes, inhaltsorientiertes Retrieval statistischer Daten mit EISREVU / Ein Verarbeitungsmodell für eine modulare Bewertung von Kennzahlenwerten für den Endanwender

Nr. 8 (1991) Schwab J.: Ein computergestütztes Modellierungssystem zur Kennzahlenbewertung

Nr. 9 (1992) Gross H.-P.: Eine semantiktreue Transformation vom Entity-Relationship-Modell in das Strukturierte Entity-Relationship-Modell

Nr. 10 (1992) Sinz E.J.: Datenmodellierung im Strukturierten Entity-Relationship-Modell (SERM)

Nr. 11 (1992) Ferstl O.K., Sinz E. J.: Glossar zum Begriffsystem des Semantischen Objektmodells

Nr. 12 (1992) Sinz E. J., Popp K.M.: Zur Ableitung der Grobstruktur des konzeptuellen Schemas aus dem Modell der betrieblichen Diskurswelt

Nr. 13 (1992) Esswein W., Locarek H.: Objektorientierte Programmierung mit dem Objekt-Rollenmodell

Nr. 14 (1992) Esswein W.: Das Rollenmodell der Organsiation: Die Berücksichtigung aufbauorganisatorische Regelungen in Unternehmensmodellen

Nr. 15 (1992) Schwab H. J.: EISREVU-Modellierungssystem. Benutzerhandbuch

Nr. 16 (1992) Schwab K.: Die Implementierung eines relationalen DBMS nach dem Client/Server-Prinzip

Nr. 17 (1993) Schwab K.: Konzeption, Entwicklung und Implementierung eines computergestützten Bürovorgangssystems zur Modellierung von Vorgangsklassen und Abwicklung und Überwachung von Vorgängen. Dissertation

Nr. 18 (1993) Ferstl O.K., Sinz E.J.: Der Modellierungsansatz des Semantischen Objektmodells

Nr. 19 (1994) Ferstl O.K., Sinz E.J., Amberg M., Hagemann U., Malischewski C.: Tool-Based Business Process Modeling Using the SOM Approach

Nr. 20 (1994) Ferstl O.K., Sinz E.J.: From Business Process Modeling to the Specification of Distributed Business Application Systems - An Object-Oriented Approach -. 1st edition, June 1994

Ferstl O.K., Sinz E.J. : Multi-Layered Development of Business Process Models and Distributed Business Application Systems - An Object-Oriented Approach -. 2nd edition, November 1994

Nr. 21 (1994) Ferstl O.K., Sinz E.J.: Der Ansatz des Semantischen Objektmodells zur Modellierung von Geschäftsprozessen

Nr. 22 (1994) Augsburger W., Schwab K.: Using Formalism and Semi-Formal Constructs for Modeling Information Systems

Nr. 23 (1994) Ferstl O.K., Hagemann U.: Simulation hierarischer objekt- und transaktionsorientierter Modelle

Nr. 24 (1994) Sinz E.J.: Das Informationssystem der Universität als Instrument zur zielgerichteten Lenkung von Universitätsprozessen

Nr. 25 (1994) Wittke M., Mekinic, G.: Kooperierende Informationsräume. Ein Ansatz für verteilte Führungsinformationssysteme

Nr. 26 (1995) Ferstl O.K., Sinz E.J.: Re-Engineering von Geschäftsprozessen auf der Grundlage des SOM-Ansatzes

Nr. 27 (1995) Ferstl, O.K., Mannmeusel, Th.: Dezentrale Produktionslenkung. Erscheint in CIM-Management 3/1995

Nr. 28 (1995) Ludwig, H., Schwab, K.: Integrating cooperation systems: an event-based approach

Nr. 30 (1995) Augsburger W., Ludwig H., Schwab K.: Koordinationsmethoden und -werkzeuge bei der computergestützten kooperativen Arbeit

Nr. 31 (1995) Ferstl O.K., Mannmeusel T.: Gestaltung industrieller Geschäftsprozesse

Nr. 32 (1995) Gunzenhäuser R., Duske A., Ferstl O.K., Ludwig H., Mekinic G., Rieder H., Schwab H.-J., Schwab K., Sinz E.J., Wittke M: Festschrift zum 60. Geburtstag von Walter Augsburger

Nr. 33 (1995) Sinz, E.J.: Kann das Geschäftsprozeßmodell der Unternehmung das unternehmensweite Datenschema ablösen?

Nr. 34 (1995) Sinz E.J.: Ansätze zur fachlichen Modellierung betrieblicher Informationssysteme - Entwicklung, aktueller Stand und Trends -

Nr. 35 (1995) Sinz E.J.: Serviceorientierung der Hochschulverwaltung und ihre Unterstützung durch workflow-orientierte Anwendungssysteme

Nr. 36 (1996) Ferstl O.K., Sinz, E.J., Amberg M.: Stichwörter zum Fachgebiet Wirtschaftsinformatik. Erscheint in: Broy M., Spaniol O. (Hrsg.): Lexikon Informatik und Kommunikationstechnik, 2. Auflage, VDI-Verlag, Düsseldorf 1996

Nr. 37 (1996) Ferstl O.K., Sinz E.J.: Flexible Organizations Through Object-oriented and Transaction-oriented Information Systems, July 1996

Nr. 38 (1996) Ferstl O.K., Schäfer R.: Eine Lernumgebung für die betriebliche Aus- und Weiterbildung on demand, Juli 1996

Nr. 39 (1996) Hazebrouck J.-P.: Einsatzpotentiale von Fuzzy-Logic im Strategischen Management dargestellt an Fuzzy-System-Konzepten für Portfolio-Ansätze

Nr. 40 (1997) Sinz E.J.: Architektur betrieblicher Informationssysteme. In: Rechenberg P., Pomberger G. (Hrsg.): Handbuch der Informatik, Hanser-Verlag, München 1997

Nr. 41 (1997) Sinz E.J.: Analyse und Gestaltung universitärer Geschäftsprozesse und Anwendungssysteme. Angenommen für: Informatik '97. Informatik als Innovationsmotor. 27. Jahrestagung der Gesellschaft für Informatik, Aachen 24.-26.9.1997

Nr. 42 (1997) Ferstl O.K., Sinz E.J., Hammel C., Schlitt M., Wolf S.: Application Objects – fachliche Bausteine für die Entwicklung komponentenbasierter Anwendungssysteme. Angenommen für: HMD – Theorie und Praxis der Wirtschaftsinformatik. Schwerpunkheft ComponentWare, 1997

Nr. 43 (1997): Ferstl O.K., Sinz E.J.: Modeling of Business Systems Using the Semantic Object Model (SOM) – A Methodological Framework - . Accepted for: P. Bernus, K. Mertins, and G. Schmidt (ed.): Handbook on Architectures of Information Systems. International Handbook on Information Systems, edited by Bernus P., Blazewicz J., Schmidt G., and Shaw M., Volume I, Springer 1997

Ferstl O.K., Sinz E.J.: Modeling of Business Systems Using (SOM), 2nd Edition. Appears in: P. Bernus, K. Mertins, and G. Schmidt (ed.): Handbook on Architectures of Information Systems. International Handbook on Information Systems, edited by Bernus P., Blazewicz J., Schmidt G., and Shaw M., Volume I, Springer 1998

Nr. 44 (1997) Ferstl O.K., Schmitz K.: Zur Nutzung von Hypertextkonzepten in Lernumgebungen. In: Conradi H., Kreutz R., Spitzer K. (Hrsg.): CBT in der Medizin – Methoden, Techniken, Anwendungen -. Proceedings zum Workshop in Aachen 6. – 7. Juni 1997. 1. Auflage Aachen: Verlag der Augustinus Buchhandlung

Nr. 45 (1998) Ferstl O.K.: Datenkommunikation. In. Schulte Ch. (Hrsg.): Lexikon der Logistik, Oldenbourg-Verlag, München 1998

Nr. 46 (1998) Sinz E.J.: Prozeßgestaltung und Prozeßunterstützung im Prüfungswesen. Erschienen in: Proceedings Workshop „Informationssysteme für das Hochschulmanagement". Aachen, September 1997

Nr. 47 (1998) Sinz, E.J.:, Wismans B.: Das „Elektronische Prüfungsamt". Erscheint in: Wirtschaftswissenschaftliches Studium WiSt, 1998

Nr. 48 (1998) Haase, O., Henrich, A.: A Hybrid Respresentation of Vague Collections for Distributed Object Management Systems. Erscheint in: IEEE Transactions on Knowledge and Data Engineering

Nr. 49 (1998) Henrich, A.: Applying Document Retrieval Techniques in Software Engineering Environments. In: Proc. International Conference on Database and Expert Systems

Applications. (DEXA 98), Vienna, Austria, Aug. 98, pp. 240-249, Springer, Lecture Notes in Computer Sciences, No. 1460

Nr. 50 (1999) Henrich, A., Jamin, S.: On the Optimization of Queries containing Regular Path Expressions. Erscheint in: Proceedings of the Fourth Workshop on Next Generation Information Technologies and Systems (NGITS'99), Zikhron-Yaakov, Israel, July, 1999 (Springer, Lecture Notes)

Nr. 51 (1999) Haase O., Henrich, A.: A Closed Approach to Vague Collections in Partly Inaccessible Distributed Databases. Erscheint in: Proceedings of the Third East-European Conference on Advances in Databases and Information Systems – ADBIS'99, Maribor, Slovenia, September 1999 (Springer, Lecture Notes in Computer Science)

Nr. 52 (1999) Sinz E.J., Böhnlein M., Ulbrich-vom Ende A.: Konzeption eines Data Warehouse-Systems für Hochschulen. Angenommen für: Workshop „Unternehmen Hochschule" im Rahmen der 29. Jahrestagung der Gesellschaft für Informatik, Paderborn, 6. Oktober 1999

Nr. 53 (1999) Sinz E.J.: Konstruktion von Informationssystemen. Der Beitrag wurde in geringfügig modifizierter Fassung angenommen für: Rechenberg P., Pomberger G. (Hrsg.): Informatik-Handbuch. 2., aktualisierte und erweiterte Auflage, Hanser, München 1999

Nr. 54 (1999) Herda N., Janson A., Reif M., Schindler T., Augsburger W.: Entwicklung des Intranets SPICE: Erfahrungsbericht einer Praxiskooperation.

Nr. 55 (2000) Böhnlein M., Ulbrich-vom Ende A.: Grundlagen des Data Warehousing. Modellierung und Architektur

Nr. 56 (2000) Freitag B, Sinz E.J., Wismans B.: Die informationstechnische Infrastruktur der Virtuellen Hochschule Bayern (vhb). Angenommen für Workshop "Unternehmen Hochschule 2000" im Rahmen der Jahrestagung der Gesellschaft f. Informatik, Berlin 19. - 22. September 2000

Nr. 57 (2000) Böhnlein M., Ulbrich-vom Ende A.: Developing Data Warehouse Structures from Business Process Models.

Nr. 58 (2000) Knobloch B.: Der Data-Mining-Ansatz zur Analyse betriebswirtschaftlicher Daten.

Nr. 59 (2001) Sinz E.J., Böhnlein M., Plaha M., Ulbrich-vom Ende A.: Architekturkonzept eines verteilten Data-Warehouse-Systems für das Hochschulwesen. Angenommen für: WI-IF 2001, Augsburg, 19.-21. September 2001

Nr. 60 (2001) Sinz E.J., Wismans B.: Anforderungen an die IV-Infrastruktur von Hochschulen. Angenommen für: Workshop „Unternehmen Hochschule 2001" im Rahmen der Jahrestagung der Gesellschaft für Informatik, Wien 25. – 28. September 2001

Änderung des Titels der Schriftenreihe *Bamberger Beiträge zur Wirtschaftsinformatik* in *Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik* ab Nr. 61

Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik

Nr. 61 (2002) Goré R., Mendler M., de Paiva V. (Hrsg.): Proceedings of the International Workshop on Intuitionistic Modal Logic and Applications (IMLA 2002), Copenhagen, July 2002.

Nr. 62 (2002) Sinz E.J., Plaha M., Ulbrich-vom Ende A.: Datenschutz und Datensicherheit in einem landesweiten Data-Warehouse-System für das Hochschulwesen. Erscheint in: Beiträge zur Hochschulforschung, Heft 4-2002, Bayerisches Staatsinstitut für Hochschulforschung und Hochschulplanung, München 2002

Nr. 63 (2005) Aguado, J., Mendler, M.: Constructive Semantics for Instantaneous Reactions

Nr. 64 (2005) Ferstl, O.K.: Lebenslanges Lernen und virtuelle Lehre: globale und lokale Verbesserungspotenziale. Erschienen in: Kerres, Michael; Keil-Slawik, Reinhard (Hrsg.); Hochschulen im digitalen Zeitalter: Innovationspotenziale und Strukturwandel, S. 247 – 263; Reihe education quality forum, herausgegeben durch das Centrum für eCompetence in Hochschulen NRW, Band 2, Münster/New York/München/Berlin: Waxmann 2005

Nr. 65 (2006) Schönberger, Andreas: Modelling and Validating Business Collaborations: A Case Study on RosettaNet

Nr. 66 (2006) Markus Dorsch, Martin Grote, Knut Hildebrandt, Maximilian Röglinger, Matthias Sehr, Christian Wilms, Karsten Loesing, and Guido Wirtz: Concealing Presence Information in Instant Messaging Systems, April 2006

Nr. 67 (2006) Marco Fischer, Andreas Grünert, Sebastian Hudert, Stefan König, Kira Lenskaya, Gregor Scheithauer, Sven Kaffille, and Guido Wirtz: Decentralized Reputation Management for Cooperating Software Agents in Open Multi-Agent Systems, April 2006