An Agent-Based Approach for Privacy-Preserving Information Filtering

vorgelegt von
Diplom-Informatiker
Richard Cissée

Von der Fakultät IV – Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des Grads

Doktor der Ingenieurwissenschaften
– Dr.-Ing. –

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Dr. rer. nat. Volker Markl
Berichter:	Prof. Dr.-Ing. habil. Sahin Albayrak
Berichter:	Prof. Dr. habil. Odej Kao

Tag der wissenschaftlichen Aussprache: 19.05.2009

Berlin 2009

D 83

Abstract

Recommender Systems and Matchmaker Systems utilize Information Filtering technologies in order to provide personalized information in the form of recommendations of items or users with similar interests, based on a user's long-term information needs, which are in turn derived from personal data and personal preferences. These systems are inherently privacy-critical because they essentially require personal data. Systems for sensitive domains in particular are not likely to be widely accepted by users unless they preserve the privacy of the user data they operate on. At the same time, Information Filtering technology providers as well as information providers have to be sufficiently motivated to develop and run privacy-friendly Recommender Systems and Matchmaker Systems. In the optimal case, these systems are multilaterally privacy-preserving in the sense that the privacy of all participating entities is preserved adequately.

This work describes an approach for distributed multilateral Privacy-Preserving Information Filtering based on Multi-Agent System technology, which due to the capabilities of agents is an obvious choice for realizing distributed privacy-preserving Recommender Systems and Matchmaker Systems. As prerequisites, we introduce mechanisms for controlling the communication capabilities of agents, which are mainly used in order to prevent agents from disclosing private data, as well as a solution for transparent persistence of data within Multi-Agent Systems, which is used in order to realize generic interactions between agents in our approach. As the core of the work, we specify two modules which allow the realization of privacy-preserving Recommender Systems as well as privacy-preserving Matchmaker Systems. The underlying agent interactions are based on cryptographic protocols, which protect participants against malicious adversaries attempting to obtain or propagate private data. We describe and examine filtering techniques that are suitable for our approach. We also describe a prototypical application based on these building blocks, and we evaluate the overall feasibility of the approach in terms of functional and non-functional requirements of privacy-preserving Information Filtering technologies.

Zusammenfassung

Empfehlungssysteme und Matchmaker-Systeme verwenden Technologien zur Informationsfilterung, um personalisierte Informationen in Form von Empfehlungen von Objekten oder Benutzern mit ähnlichen Interessen zu liefern. Diese Empfehlungen basieren auf langfristigen Informationsbedürfnissen eines Benutzers, welche aus den persönlichen Daten und persönlichen Präferenzen dieses Benutzers abgeleitet werden. Aufgrund dieses expliziten Bedarfs an persönlichen Daten ist in derartigen Systemen die Privatheit der Benutzer inhärent bedroht. Insbesondere in sensiblen Domänen werden diese Systeme wahrscheinlich nur auf breite Akzeptenz stoßen, wenn sie die Privatheit der zugrundeliegenden Benutzerdaten bewahren. Gleichzeitig müssen die Anbieter von Technologien zur Informationsfilterung und die Anbieter der zugrundeliegenden Informationen hinreichend motiviert sein, privatheitsunterstützende Empfehlungssysteme und Matchmaker-Systeme zu entwickeln und zu betreiben. Optimalerweise berücksichtigen derartige Systeme die Privatheit aller beteiligten Parteien und sind somit mehrseitig privatheitsbewahrend.

Diese Arbeit beschreibt einen Ansatz zur verteilten mehrseitigen privatheitsbewahrenden Informationsfilterung, basierend auf Multiagentensystem-Technologie. Aufgrund der Fähigkeiten von Agenten liegt es nahe, diese Technologie zur Umsetzung von verteilten Empfehlungssystemen und Matchmaker-Systemen einzusetzen. Als Grundlagen beschreibt diese Arbeit Mechanismen zur Einschränkung der Kommunikationsfähigkeiten von Agenten und eine Lösung für transparente Persistenz von Daten in Multiagentensystemen. Erstere werden verwendet, um Agenten an der unkontrollierten Weitergabe von privaten Daten zu hindern, während letztere verwendet wird, um generische Interaktionen zwischen den beteiligten Agenten zu ermöglichen. Den Kern der Arbeit stellt die Spezifikation zweier Module dar, welche die Umsetzung von privatheitsbewahrenden Empfehlungssystemen und Matchmaker-Systemen ermöglichen. Die zugrundeliegenden Interaktionen zwischen Agenten basieren auf kryptographischen Protokollen, welche von die beteiligten Parteien zum Schutz vor böswilligen Angriffen verwendet werden, in denen versucht wird, private Daten zu erhalten oder weiterzuverbreiten. Die Arbeit beschreibt und bewertet Filterverfahren, die unter den gegebenen Bedingungen einsetzbar sind. Weiterhin beschreibt die Arbeit eine prototypische Anwendung, welche auf diesen Modulen aufbaut, und evaluiert abschließend den gesamten Ansatz, basierend auf den funktionalen und nichtfunktionalen Anforderungen von Technologien zur privatheitsbewahrenden Informationsfilterung.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The total quantity of information stored in electronic format is increasing exponentially [80]. Consequently, human users accessing information are increasingly threatened by information overload, i.e. by the fact that the amount of information available on a specific topic cannot be handled manually any more.

Information Retrieval (IR) technologies enable users to specify short-term information needs e.g. via search engines, and receive a manageable amount of relevant information meeting their information needs. In many cases, however, users have long-term information needs which are only met inadequately by the use of IR technologies because the respective information changes dynamically. As an example, if a user intends to keep track of new publications within a specific area, he would have to specify the same information need over and over again when using a search engine, and he would have to filter the results manually in order to determine which results actually have not been received before.

Information Filtering (IF) technologies, as the name implies, are a solution for this problem, because they enable users to specify long-term information needs and to receive the respective information in an manageable manner: Utilizing IF-based Recommender Systems or Matchmaker Systems, a user is provided with probably relevant information in the form of recommendations (e.g. of documents or other users) that are determined based on information known about the user. This personal information is stored in a user profile and may contain the user's general preferences, ratings of information obtained previously, or references to other users with similar interests.

Shifting the task of filtering relevant information from the user to a software system effectively eliminates the threat of information overload, but unfortunately introduces threats with regard to the privacy of the user: While the use of IR technologies such as search engines is generally not privacy-critical because single queries cannot be used to obtain information about a user[1], the use of IF technologies is inherently privacy-critical because these technologies explicitly require personal data.

This apparent discrepancy of personalization and privacy has to be overcome if solutions based on IF technologies are to be widely accepted. While users may be less concerned about their privacy in domains covered by existing Recommender Systems used e.g. in e-commerce applications, Recommender Systems for more sensitive domains providing e.g. financial or health-related information are not likely to be accepted unless they address privacy threats. At the same time, providers have to be motivated sufficiently to offer privacy-friendly Recommender Systems, which they may initially regard as problematic because these systems cause customer data to be less accessible.

While there are several candidates for technologies which may be used to realize a solution for privacy-preserving information filtering, this work shows that Multi-Agent System technology is ideally suitable for realizing a distributed system with the given requirements. Therefore, the solution described in this work does not merely constitute the realization of an abstract specification in the context of a Multi-Agent System (MAS) system, but rather an approach which essentially requires the features provided by MAS technology.

We summarize these aspects as the main motivation for this work:

- The main goal of this work is to provide a solution for Privacy-Preserving Information Filtering in which the aspects of personalization and privacy are no longer irreconcilable. Users intending to receive personalized information should not have to give up their privacy with regard to the personal data this information is based on.

- The solution should not preserve the privacy of the users at the cost of reducing the privacy of the other participants, such as the information provider. Rather, it should address the privacy of all participants and thus achieve *multilateral privacy*.

- In addition to these goals, the work is also motivated by the idea of realizing a solution which highlights the capabilities of Multi-Agent System

[1]The privacy of users is at risk, however, when it is possible to link larger numbers of queries to a single user, even when the respective user is not directly identifiable. As a recent example, when AOL released 20 million pseudonymized search engine queries from several hundreds of thousands of users in 2006, it was possible to reconstruct the identities and information needs of users based on their queries.

technology and shows the potential benefits of a wider propagation of this technology.

1.1 Main Contributions

As its title implies, this work describes an approach for Privacy-Preserving Information Filtering, which consists of five main parts as the main contributions of this work:

- We introduce mechanisms for controlling the communication capabilities of agents, which are mainly used in order to prevent agents from disclosing private data.
- We introduce a mechanism for transparent persistence of data within a MAS, which is used in order to realize generic interactions between participants in our approach.
- We specify interactions and protocols used to realize privacy-preserving Recommender Systems.
- We specify interactions and protocols used to realize privacy-preserving Matchmaker Systems, i.e. systems determining users with similar interests.
- We examine and describe filtering techniques that are suitable for our approach.

Work and results related to this thesis has been published in the following papers and articles:

- R. Cissée. An Architecture for Agent-Based Privacy-Preserving information filtering. In *Proceedings of the 6th International Workshop on Trust, Privacy, Deception and Fraud in Agent Systems*, 2003.[31].
- R. Cissée and S. Albayrak. An Agent-Based Approach for Privacy-Preserving Recommender Systems. In E. H. Durfee, M. Yokoo, M. N. Huhns, and O. Shehory, editors, *6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007), Honolulu, Hawaii, USA, May 14–18, 2007*, IFAAMAS, 2007.[32].
- J. Wohltorf, R. Cissée, and A. Rieger. Berlintainment: An Agent-Based Context-Aware Entertainment Planning System. *IEEE Communications Magazine, 43(6)*, 2005.[117].

The first publication mainly contains the initial ideas of this work. The second publication summarizes the results of this thesis, focusing on the central aspects of communication control and Recommender System functionality. The third publication primarily describes the prototypical application in which our solution for PPIF has been utilized.

1.2 Structure of the Thesis

This thesis consists of the following four main parts, as shown in Table 1.1:

Table 1.1: Overview of the structure of the Thesis.

Part		Chapter	
I	context	1	Introduction
		2	Problem Description
		3	Related Work
II	approach	4	Privacy-Preserving Information Filtering
		5	Basic Infrastructure
		6	Transparent Persistence
		7	The Recommender Module
		8	The Matchmaker Module
		9	Exemplary Filtering Techniques
III	review	10	Evaluation
		11	Conclusion & Outlook
IV	details	A	Specification of Ontologies, Roles and Interactions
		B	Basic Infrastructure: Examples
		C	Exemplary Filtering Techniques: Examples

- The first part provides the context for our work. It comprises this chapter as the introduction and the two following chapters: Chapter 2 describes the problems and states the requirements of Privacy-Preserving Information Filtering (PPIF), by addressing the two areas of privacy and Information Filtering separately and in combination. It provides definitions used throughout the work, and lists requirements for PPIF. As our solution for PPIF is realized via Multi-Agent System (MAS) technology, it also provides definitions used in this context, and describes the central problem of malicious hosts in MAS systems. Chapter 3 reviews related work in the areas of Privacy-Enhancing Technologies and Privacy-Preserving Technologies, and the state of the art in privacy-preserving Information Filtering.

- The second and main part describes our approach. It comprises a chapter outlining the overall approach and implemention, and five chapters describing the main components of our approach: Chapter 4 lists the supported use cases and gives a high-level outline of our solution for PPIF, based on two essential concepts, namely the concept of a trusted environment for protecting privacy in Recommender Systems, and the additional concept of an anonymous centralized model for protecting privacy in Matchmaker Systems. It motivates the use of MAS technology in this context, and describes the implementation of the approach itself as well as the implementation of a prototypical application based on the approach. Chapter 5 describes basic functionality for controlling the communication capabilities of agents, and functionality for anonymous communication of agents. Chapter 6 describes an approach for Transparent Persistence in Multi-Agent Systems (TPMAS). Chapter 7 describes functionality for realizing Recommender System functionality. Chapter 8 describes functionality for realizing Matchmaker System functionality. Chapter 9 describes exemplary filtering techniques that are applicable in this context.

- The third part constitutes a review our approach. It comprises the following two chapters: Chapter 10 evaluates our approach for Privacy-Preserving Information Filtering. It discusses the coverage of the non-functional and functional requirements by our approach and by the implemented application. It also compares our approach with approaches based on trusted software, and provides usage guidelines for applying the functionality specified in our approach. Chapter 11 concludes the work by discussing the applicability of the approach in large-scale real-world applications and by discussing directions for further research.

- The forth part provides details of our approach. It comprises the following three appendices: Appendix A provides tables and diagrams containing the formal specification of the components of our approach for Privacy-Preserving Information Filtering. Appendix B illustrates the interactions for controlling communication. Appendix C provides a brief example for the algorithm used by a suitable filtering technique, namely hierarchical agglomerative clustering via single-link clustering.

By mapping each main contribution of this work to the respective chapter of the main part of the thesis, the structure of the second part directly represents the main contributions of this work.

1.3 Methodology and Notation Conventions

Analysis and design of the main components of our approach is kept as generic as possible. We follow the Gaia methodology [120] for Agent-Oriented Software Engineering and use AUML [11] as the modeling language for diagrams. As utilizing all steps of the Gaia methodology in this work would be neither illustrative nor practicable, we concentrate on the essential steps, i.e. mainly the analysis and design phases.

1.3.1 Analysis Section

Each analysis section combines the analysis and architectural design phase of the Gaia methodology. For every module, the analysis section contains the ontologies in which the required domain knowledge is conceptualized and specified. An ontology consists of categories and ontology functions which may be applied to these categories. Categories contain attributes, whereas each attribute has type, which may be a base type or a category type. Categories may inherit attributes from other categories. For the notation of ontologies, we use AUML-based diagrams.

Furthermore, the analysis section contains the role model describing the participating roles and interactions. Roles are denoted as ExemplaryRole. The responsibilities of a role are specified as lifeness expressions exemplary-Role$_i$ with $i = 1, .., n$. Internal activities are denoted as $\underline{\mathsf{ExemplaryActivity}}$. Interactions are denoted as $\mathsf{ExemplaryInteraction}$. An interaction consists of single communication steps as protocol parts, denoted as int_{si} for the sender role and int_{ri} for the receiver role in the interaction Int, with $i = 1, .., n$. However, when specifying an interaction, we usually abstract from single communication steps in the overall protocol and just indicate the following three protocol parts:

$$\mathsf{Int}_I \overset{def}{=} \mathsf{int}_{s1}.\mathsf{int}_{rn}$$
$$\mathsf{Int}_R \overset{def}{=} \mathsf{int}_{r1}$$
$$\mathsf{Int}_P \overset{def}{=} \mathsf{int}_{sn}$$

Interactions may be interleaving. As an example, the lifeness expressions

$$\begin{aligned}
\textsc{firstRole} &= \mathsf{FirstInt}_I \\
\textsc{secondRole} &= \mathsf{FirstInt}_R.\mathsf{SecondInt}_I.\mathsf{FirstInt}_P \\
\textsc{thirdRole} &= \mathsf{SecondInt}_R.\mathsf{SecondInt}_P
\end{aligned}$$

imply that the actual communication steps may be carried out as follows:

$$\begin{aligned}
\textsc{firstRole} &= \mathsf{firstInt}_{s1}.\mathsf{firstInt}_{r2}.\mathsf{firstInt}_{s3}.\mathsf{firstInt}_{r4} \\
\textsc{secondRole} &= \mathsf{firstInt}_{r1}.\mathsf{secondInt}_{s1}.\mathsf{secondInt}_{r2}.\mathsf{firstInt}_{s2}. \\
&\quad \mathsf{firstInt}_{r3}.\mathsf{secondInt}_{s3}.\mathsf{secondInt}_{r4}.\mathsf{firstInt}_{s4} \\
\textsc{thirdRole} &= \mathsf{secondInt}_{r1}.\mathsf{secondInt}_{s2}.\mathsf{secondInt}_{r3}.\mathsf{secondInt}_{s4}
\end{aligned}$$

Interactions may be anonymous. A mechanism for anonymous interaction realizing anonymity of the initiator as well as unlinkability of the single communication steps is described in Chapter 5. A protocol part in which the respective participant remains anonymous is indicated as Int_I^{anon}. If the interaction consists of more than the minimal two communication steps, subsequent pairs of communication steps may be assumed to be unlinkable.

Interactions are visualized via AUML-based collaboration diagrams. Here, instances of a specific role are denoted as "id:RoleName", or as "id@platform-id:RoleName", if the platform configuration for the deployment of the agents realizing the respective roles is relevant.

1.3.2 Design & Implementation Section

Each design section addresses the steps of the detailed design phase of the Gaia methodology. Based on the role model, an agent model is specified. Agents are denoted as **ExemplaryAgent**. Agent services are denoted as *ExemplaryAgentService*. In most cases, the mapping of roles to agents and interactions to agent services is rather straightforward and therefore not described in detail. Details of the implementation of the parts of our approach that are realized as agent services are omitted for the same reason. Instead, we focus on implementation aspects only when the respective functionality has not been specified in terms of ontologies, roles, and interactions as described above.

1.3.3 Cryptographic Protocols

For the cryptographic protocols used within our approach, a standard notation following [20] is used where each protocol step is written as

$$n\colon\ S \rightarrow R\colon\ m$$

where n indicates the protocol step, S and R two principals, namely the sender and receiver, and m the message.

Steps that may be carried out in parallel are denoted as na, nb, etc. Anonymous communication is denoted as $n^{anon(S)}$ and $n^{anon(R)}$ for sender and receiver anonymity respectively. A message contains data $X = \{x_1, .., x_p\}$, which is often processed by a specific function f (such as an encryption function), i.e. $m = f(X)$. A protocol step may be carried out iteratively for the partial data of a message, in order to achieve unlinkability of the data. This is denoted as

$$\forall x \in X\colon\ n_x\colon\ S \rightarrow R\colon\ f(x).$$

If there are subsequent iterations on the same data, as in the following example

$$\begin{aligned} &\forall x \in X\colon n_x\colon\ A \rightarrow B\colon\ f(x) \\ &\forall x \in X\colon (n{+}1)_x\colon\ B \rightarrow C\colon\ g(x), \end{aligned}$$

the step $(n+1)_{x_i}$ may be carried out as soon as the step n_{x_i} is finished, without the complete step n being required to have been completed.

Encryption functions are denoted as follows: The term $f_K(x)$ denotes a statement x encrypted with a key K under a specific encryption function f. If f is implicitly given, we use $\{x\}_K$ as a short expression. The term $f'_K(x)$ denotes a encrypted statement x decrypted with a key K under a specific decryption function f'. The protocols in this work are largely based on symmetric encryption schemes, i.e. schemes where a single key is used for both encryption and decryption[2], so that $f'_K(\{x\}_K) = x$. In asymmetric encryption schemes, a public key K is used for encryption, and a private key K' for decryption, where the private key cannot be derived from the public key in a feasible manner, so that $f'_{K'}(\{x\}_K) = x$. Conversely, a digital

[2]The actual key used for decryption may differ from the actual key used for encryption as long as the keys are trivially related.

signature s of statement x is usually obtained via encryption of its hash with a private key, i.e. $s_{K'}(h(x)) = \{h(x)\}_{K'}$. It is verifiable via the public key K as $f'_K(\{h(x)\}_{K'}) = h(x)$.

We denote the encryption of a set of statements $X = \{x_1, .., x_p\}$ as $\{X\}_K = (\{x_1\}_{K_{x_1}}, .., \{x_p\}_{K_{x_p}})$. The keys K_{x_1} to K_{x_p} may actually be the same key, but different keys may have to be used depending on the protocol and the encryption scheme. A secret key known only to principal A at the start of the protocol is written as K_A. A key shared between two principals A and B is written as K_{AB}.

An encrypted hash of message m is denoted as $h(m)$, where h denotes a *cryptographic hash function*, i.e. a function returning a fixed-size string for a message of arbitrary length with the following additional properties:

- *Preimage resistant*: For a given $h(m)$ and unknown m, it should be hard (i.e. computationally infeasible) to find messages m' with $h(m') = h(m)$.
- *Second preimage resistant*: For a given m, it should be hard to find messages m' with $h(m') = h(m)$.
- *Collision-resistant*: It should be hard to find any two messages m and m' with $h(m') = h(m)$.

In order to be able to verify both the integrity and the authenticity of a message, a keyed-Hash Message Authentication Code (HMAC) may be generated, based on a cryptographic hash function and a secret key, via $HMAC_K(m) = h(K^* + h(K^* + m))$ with K^* being the key K padded with extra zeroes as required by the hash function. Analogous to using $\{m\}_K$ as a short expression for an encrypted message, we use $\{h(m)\}_K$ as a short expression for a HMAC. We also use $H(M)$ as a short expression for $\{h(m_1), h(m_2), .., h(m_n)\}$ with $M = \{m_1, m_2, .., m_n\}$. Note that $H(M) \neq h(M)$, because the latter expression denotes a single hash of a set of messages, while the former expression denotes a set of hashes of single messages.

Chapter 2

Problem Description

This work deals with privacy in the context of Information Filtering (IF). In order to be able to describe problems and state the requirements of Privacy-Preserving Information Filtering (PPIF), we first address these two areas separately, and provide definitions used throughout the work. Subsequently, we examine the intersection of the two areas and list requirements for PPIF. As our approach for PPIF is realized via Multi-Agent System (MAS) technology, we also provide definitions used in this context, and describe the central problem of malicious hosts in MAS systems. Thus, this chapter is structured as follows: The following section deals with privacy in general. Section 2.2 introduces concepts, definitions and architectures related to of Information Filtering. Section 2.3 discusses the aspect of privacy in the context of Information Filtering architectures, and states the requirements for Privacy-Preserving Information Filtering. Section 2.4 deals with MAS technology. Section 2.5 summarizes the results of this chapter.

2.1 Privacy

In this section, we establish a definition of privacy that is used throughout this work, and we give a short overview of different strategies for protecting privacy, one of which we follow in this work.

2.1.1 Definitions

The term *privacy* describes a fundamental human right recognized by societies worldwide [43]. There is, however, no single universally accepted definition of privacy. Many different definitions of privacy have been proposed, mainly because the concept of privacy encompasses different aspects whose importance cannot be quantified objectively, leading to either very broad or extremely narrow definitions. Furthermore, different definitions arise from attempts to distinguish between the related concepts of privacy, confidentiality, and secrecy. In the following, we restrict the discussion to largely accepted definitions that are most suitable in the context of this work.

The following definition is given by political scientist Alan Westin: "[Privacy is] the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others" [116]. It is extended and complemented by the definition given by psychologist Roger Ingham: "Privacy is concerned with the claim that individuals or groups have to determine for themselves how, when and to what extent certain aspects of their behavior are determined by others, behavior in this context being generously defined" [63]. Taken together, these definitions encompass most aspects of privacy. They agree on the fact that privacy is related to the ability to create barriers between individuals or groups on the one hand and the society or parts thereof on the other hand. They focus, however, on different aspects of these barriers: They may be used in order to prevent the dissemination of information (according to Westin's definition), i.e. as *active* privacy protection, or in order to protect against intrusions (according to Ingham's definition), i.e. as *passive* privacy protection. In the first case, the initiative originates from the individual or group trying to expand or at least maintain a barrier, while in the latter case the initiative is taken by another party trying to breach a barrier.

There are two further aspects of privacy, which are largely orthogonal to the aspects described above:

- *Physical privacy* is defined as "freedom from unauthorized intrusion" [83], or, more precisely, as "a restriction on the ability of others to experience a person through one or more of the five senses" [88]. It has been recognized as a basic right by future Supreme Court justice Brandeis as early as 1890 [115], defining it as the "right to be let alone" in 1928 [105]. These definitions show that physical privacy is more closely related to passive privacy.

- *Informational privacy* is often merged with concept of data protection. It is best defined as a right to informational self-determination, authorizing "each individual to determine on the circulation and the use of his own personal data"[1] [18], thus controlling the dissemination of personal information. While sometimes a distinction is made between

[1]Translated from the definition "die Befugnis des Einzelnen, grundsätzlich selbst über die Preisgabe und Verwendung seiner persönlichen Daten zu bestimmen."

individuals and organizations with regard to this aspect, it is in fact reasonable to apply this definition to groups and organizations as well, as in the definitions of Westin and Ingham given above. Another aspect of informational privacy is the "protection against intrusion by unwanted information" [87]. Informational privacy is therefore more closely related to active privacy.

Based on the given definitions, we define four aspects of privacy as shown in Figure 2.1. The combinations of these aspects are roughly equivalent to the four aspects of privacy as defined by Westin [116]:

- *Solitude*, i.e. the separation from direct or indirect observation and interference by others, is related to the passive aspect of physical privacy.
- *Intimacy*, i.e. the ability to keep physical interactions and communication private, is related to the active aspect of physical privacy.
- *Reserve*, i.e. the decision not to reveal certain information to others, is related to the active aspect of informational privacy.
- *Anonymity*, i.e. The avoidance of identification, is related to the passive aspect of informational privacy. This relation, however, is less close than the relations regarding the previous aspects.

	physical privacy	*informational privacy*
active privacy	Private Interactions/ Communications (INTIMACY)	Informational Self-Determination (RESERVE)
passive privacy	"the right to be let alone" (SOLITUDE)	Avoidance of Identification (ANONYMITY)

Figure 2.1: Aspects of privacy.

In this work, we focus on informational privacy and disregard aspects of physical privacy. Therefore, in the following the term "privacy" always denotes informational privacy.

2.1.2 Strategies for Protecting Privacy

The active protection of informational privacy is a non-trivial task, because controlling the dissemination of information is almost impossible in scenarios that require more complex actions than all-or-nothing approaches: Allowing unrestricted access to information as the one extreme, or withholding information entirely as the other extreme are approaches that may be accomplished in a straightforward manner. However, these approaches are insufficient in cases where the respective information actually has to be propagated to certain other parties, but at the same time is intended to be kept under some control with regard to further dissemination.

The main problem therefore is the following: How may a party prevent further dissemination of information once it has lost the exclusive control over this information? There are three main strategies addressing this problem:

- *Legal Regulation*: In most developed countries, legal regulations exist that deal with the protection of privacy. As examples, we review the current situation in the United States of America and the European Union.

  In the United States, privacy as a fundamental human right is not explicitly addressed in the Constitution. Specific aspects of privacy, however, are protected implicitly, e.g. by the Fourth Amendment (addressing the passive aspect of physical privacy), or the First Amendment (addressing the active aspect of privacy). Currently the main foundation for data protection and informational privacy in the United States are the Principles of Fair Information Practices based on the Code of Fair Information Practices [109] and further refined in OECD guidelines [89]. They are agreed upon by industry groups, privacy experts and the United States government. The Fair Information Principles are intended to limit data collection and to enable individuals to control the dissemination of their personal information.

  In the European Union, the key document addressing privacy is Directive 95/46/EC of the European Parliament, known as the European Union Privacy Directive or European Union (EU) Data Protection Directive [40], which binds member states of the European Union to ratify laws implementing its requirements. The directive restricts the collection and processing of personal information, and prohibits the transfer of personal information to entities for whom less strict legal regulations apply, such as companies in non-member states. Additionally, the *safe harbor* framework has been introduced to allow the transfer of personal data between the EU and the United States, comprising seven principles consistent with the Principles of Fair Information Practices. Some member states of the EU have introduced more extensive privacy and data protection regulations.

- *Self-Regulation*: Today, the processing of personal information collected via the internet, i.e. by web sites accessed by users in the context of e-Commerce, is mainly regulated by *privacy policies* posted on the respective site. These privacy policies are generally not legally binding and therefore constitute mechanisms within the area of self-regulation. Comprehensive privacy policies should, according to the Fair Information Principles, address at least the following principles:

  – *Notice/ Awareness*: The user has to be noticed of the provider's privacy policy before any information about the user is collected. If the user is not aware of the provider's privacy policy, the following principles are meaningless.

  – *Choice/ Consent*: The user has to be given the possibility to explicitly decline the collection and further dissemination of personal information, i.e. the ability to *opt-out*, or preferably the possibility to explicitly allow these actions, i.e. the ability to *opt-in*.

  – *Access/ Participation*: The user has to be given means to access all personal information collected about him and has to be able to change or delete at least information that is inaccurate.

  – *Security/ Integrity*: The collected personal information should be stored securely without third parties being able to obtain it. Additionally, the integrity of the information should be ensured.

  – *Enforcement/ Redress*: The provider should act according to his stated privacy policy. If there is no enforcement mechanism in place (by legal regulation, external audits, certification or other means), the privacy policy is merely an assertion of the provider rather than an agreement between the two parties involved.

Privacy seal programs have been introduced in order to inform users of the enforcement of privacy policies. Compared to standard privacy policies, they have two advantages: The user may decide immediately whether he wants to interact with a provider, based on the displayed seals. If the user trusts the seal issuer, he does not have to deal with the respective privacy policy itself, which is in many cases rather complicated and hard to interpret for users who are not familiar with legal terminology. Furthermore, the user probably trusts an independent seal issuer more readily than he would trust a single, perhaps formerly unknown provider. It has been pointed out, however, that current seal programs generally do not enforce sanctions against malicious providers and are therefore largely ineffective [33].

- *Technology*: A wide range of Privacy-Enhancing Technologies and Privacy-Preserving Technologies have been suggested and implemented, addressing different aspects of privacy in the context of various applications. These approaches are discussed in detail in the following chapter.

For all practical purposes, all three approaches ultimately require some amount of trust: The user whose privacy is to be protected has either to trust in the responsible legislative, executive and judicial offices to actually enforce legal regulations, prosecute violations, and abide by these regulations; or he has to trust in the parties he interacts with to adhere to the stated self-regulatory principles; or he has to trust in the technology that is applied to work as specified. However, strategies based on technology may (at least theoretically) not require trust, because the technologies used my be examined and verified by the user (although in practice most users do not have the required knowledge to do so). In this work we focus on privacy protection through technology and therefore subsequently ignore the other strategies.

2.2 Information Filtering

In this section, we provide definitions that are relevant in the context of Information Filtering, we discuss the main types of IF architectures, and we list the main problems of IF architectures.

2.2.1 Definitions

In the following, the definitions of the most important terms related to Information Filtering are given.

- The term "Personalization" in its broadest definition denotes the adaption of products or services to individual consumers or users, based on the acquisition and analysis of personal information, i.e. information related to a single specific person. It is used mainly in the context of Web

Personalization (i.e. in the context of personalized web sites and personalized content provided via the World Wide Web) [70], but the underlying concepts have been widely researched under the designation "User Modelling and User-Adaptive Interaction" [71]. Related terms often used synonymously are "Customer Relationship Management (CRM)" and "1-to-1 Marketing".

- The term "Information Filtering" denotes concepts and methods used to provide personalized information. Systems dealing with other aspects of personalization, such as personalized user interfaces, do not constitute IF systems. IF differs from the related field of Information Retrieval in the following aspect: While the objective of IR systems, such as search engines, is the fulfillment of short-term *information needs*, IF systems deal with long-term information needs. Therefore, while IR systems are based on queries expressing an ephemeral information need, IF systems are usually based on user profiles expressing persistent information needs. An IF system may contain IR functionality, e.g. for refining a long-term information need in a specific context, resulting in a combination of both kinds of information needs. An exemplary scenario is described in 4.1. Moreover, Information Filtering is distinguished from the related field of *Web Mining* by the following definitions: While Web Mining systems search for relevant information across multiple web resources, IF systems operate on structured information already at hand.

- The term "Recommender System" (synonymous with "Recommendation System") originally denoted a system for *Automated Collaborative Filtering* [96]. The definition, however, has been broadened and now denotes "any system that produces individualized recommendations as output or has the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options" [19].

- The term "Matchmaker System" (synonymous with "Matchmaking System") denotes a system aiming at introducing similar users to each other [45]. In this context, similarity is usually determined via personal user profiles which contain general preferences and additional data indicating a user's interests. It should be noted that especially in the context of MAS architectures, the term sometimes denotes a different kind of system, namely a system assisting users or agents in finding and accessing certain resources [74]. We do not follow this definition in this work.

According to these definitions, an IF system can be classified as a Recommender System, a Matchmaker System, or as a Hybrid IF System, i.e. a combination of both: A Matchmaker System may be combined with a Recommender System, usually by generating personalized information via similar users. A pure Recommender System may also determine similar users as an intermediate step in the process of generating personalized information, but does not introduce similar users to each other. In other words, it is possible to generate personalized information via similar users, but it is not always possible to determine similar users via personalized information. Regarding Matchmaker Systems and Hybrid IF Systems, we additionally distinguish between centralized and distributed approaches, depending on the prevalent mode of interaction, as described below. Figure 2.2 shows the relation between these systems and the related areas.

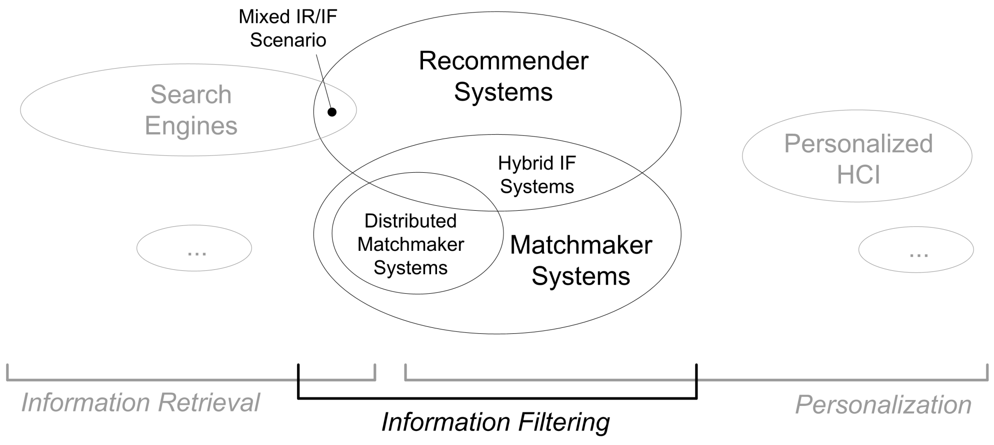


Figure 2.2: A classification of Information Filtering systems (in black) and related areas (in gray).

An IF system consists of three main kinds of abstract entities:

- The *user*, who intends to receive personalized information. Additionally, users may provide feedback in order to improve the quality of the IF system. The user entity directly or indirectly (e.g. via a user agent) represents a human user[2]. In distributed IF systems, the user entity also directly participates in the process of generating personalized information for a specific, different user entity.

[2]In some systems, additional artificial user entities are introduced: It has been suggested, for example, to utilize software agents in order to simulate human users with the goal of improving the quality of recommendations by creating additional ratings [59].

- The *provider*, who provides the information based on which ultimately personalized information is generated, and from which user profile items are obtained. The provider entity usually represents a legal entity, such as a company. In non-distributed IF systems, the provider entity also directly participates in the process of generating personalized information for a specific user entity.
- The *filter*, who provides *filtering techniques* as collections of algorithms for generating personalized information. The filter entity usually represents a legal entity, such as a company.

Table 2.1 gives an overview of these and further terms used in the following formal definitions.

Table 2.1: Terminology used in the context of Information Filtering systems throughout this work.

Notation	Term
u	user entity
U	the set of all user entities
p	provider entity
f	filter entity
s	supplier entity, $s \in \{u, p\}$
i	item
I	a set of items
PR	a profile
$q_{IR}(PR)$	constrained profile (as result of IR-based query on a profile)
$q_{IF}(PR)$	partial profile (as result of IF-based query on a profile)
m	a profile model
ft	filtering technique
FT	the set of all filtering techniques
pp	filtering technique algorithm (used during the information processing stage)
ff	filtering technique algorithm (used during the information filtering stage)
$pred$	a prediction of relevance
REC	a set of recommendations
SU	a set of similar user entities
RES	a set of results (recommendations or similar user entities)

Because users generally do not want to re-enter personal information for every single filtering process, and the amount of provider information often prohibits the direct application of a filtering technique to large collections of data, an IF system consists of two additional stages preceding the information filtering stage itself. Thus, we define three separate stages in an IF system:

- *Information Collection Stage*: In the first stage, the data to be used as the input for subsequent filtering processes is collected. This stage is independent of the actual filtering technique. On the user side, a *user profile* is generated and maintained based on interaction with the user and/or observation of the user's actions: The user profile contains profile elements, which are *items* (provided by the provider or a different source) that have been rated either explicitly by the user himself, or implicitly by analyzing the user's behavior. Items usually contain pairs of attributes and values. The user profile may also contain general preferences of the user, or additional information, such as personally identifying information and demographic data. On the provider side, a *provider profile* is provided (via an unspecified mechanism) containing a (usually large) collection of items of a specific domain. The formal definition of a profile is given in Equation 2.1. As entities supplying profiles, both user entity and provider entity are subsumed as *supplier* entities.

- *Information Processing Stage*: In the second stage, the data collected in the first stage is processed further as a preparation for the final stage. Based on the profile elements and an actual filtering technique, *models* are created on the user and provider side. These models constitute an additional part of the respective profile, which is therefore defined as described by Equation 2.1.

$$PR_s = I_s \cup \bigcup_{ft \in FT} m_{I_s,ft} \tag{2.1}$$

  Simple filtering techniques may use the profile elements themselves as input (on one side or on both sides), but in most cases models are generated and maintained in order to reduce the complexity of the final stage, or because the underlying algorithm explicitly requires a model. In any case, the structure of the model depends on the filtering technique to be applied. Exemplary models are neural networks, data clusters, and decision trees. Based on the way the models are generated and maintained, we distinguish between two main groups of filtering techniques:

  - In *feature-based approaches* the models are generated and maintained separately for the user and provider side, based solely on the features of the respective profile items. Data is not exchanged between different profiles. Therefore, feature-based models are defined as described by Equation 2.2.

$$m_{I_s,ft} = pp_{ft}(I_s) \tag{2.2}$$

  - In *collaboration-based approaches* the models are generated and maintained via a collaboration of entities, usually different user entities and a provider entity. Provider models therefore contain user profile data, and user models may contain provider profile data as well. Therefore, collaboration-based models are defined as described by Equation 2.3 and Equation 2.4.

$$m_{I_u,ft} = pp_{ft}(I_u, I_p) \tag{2.3}$$

$$m_{I_p,ft} = pp_{ft}(I_p, \bigcup_{u \in U} I_u) \tag{2.4}$$

- *Information Filtering Stage*: In the third and final stage, the filtering technique is applied to two profiles (either to the profile items or to models) in order to generate personalized information. The personalized information is always generated for a user. The information the personalized information is based on is supplied by a provider entity or a different user entity, i.e. $s = p$ or $s = u''$. A process of this stage generates one of the following four main kinds of results:

  - A *prediction* of the relevance of a specific item for the given user. In this case, the item in question has to be provided as additional input to the filtering technique, and a filtering algorithm as defined in Equation 2.5 is required. The supplier is the provider (in a Recommender System or a non-distributed Hybrid IF System) or a different, presumably similar user (in a distributed Hybrid IF System). The prediction values are usually defined as a range of possible values, where a higher value indicates a higher predicted relevance. Additionally, the value reflects the relevance in relation to other candidate items available to the supplier. Otherwise, a supplier profile would not actually be required by the filtering algorithm.

$$pred_{u,s,ft,i} = ff_{ft}(PR_u, PR_s, i) \tag{2.5}$$

- The *top-n recommendations* for a given user, i.e. a set of n items (or less, e.g. in case the number of candidate items is smaller than n) that are recommended to the given user because they have the highest predicted relevance of all candidate items available to the supplier that are not already contained within the user profile. In this case, a filtering algorithm as defined in Equation 2.6 is required. The supplier is the provider (in a Recommender System or a non-distributed Hybrid IF System) or a different, presumably similar user (in a distributed Hybrid IF System).

$$\begin{aligned} REC_{u,s,ft,n} = ff_{ft}(PR_u, PR_s, n) = \\ \{i \in (PR_s \setminus PR_u) | \, \forall \, X \subseteq (PR_s \setminus PR_u \setminus \{i\}) : \\ |X| < n \quad \vee \quad \exists \, x \in X : pred_{u,s,ft,i} > pred_{u,s,ft,x}\} \end{aligned} \tag{2.6}$$

  In case of large supplier profiles, a model-based filtering algorithm may be used instead, leading to results that may differ somewhat from this definition.

- A *prediction* of the similarity of a specific user and the given user. Similarity may be based on users' features, preferences, or any kind of data related to users. The user in question has to be provided as an additional input to the filtering technique. In this case, a filtering algorithm as defined in Equation 2.7 is required. The supplier is the provider (in a Matchmaker System) or a different user (in a distributed Matchmaker system), which may either be a candidate that is likely to be similar itself (i.e. $s = u'' = u'$) or a different user (i.e. $s = u'' \neq u'$). For statements applying to the prediction values, see above.

$$pred_{u,s,ft,u'} = ff_{ft}(PR_u, PR_s, u') \tag{2.7}$$

- The *top-n similar users* for a given user, i.e. a set of n users (or less, e.g. in case the number of candidate users is smaller than n) that are returned to the given user because they have the highest predicted similarity value of all candidate users available to the supplier. In this case, a filtering algorithm as defined in Equation 2.8 is required. The supplier is the provider (in a Matchmaker System) or a different, presumably similar user (in a distributed Matchmaker system).

$$\begin{aligned} SU_{u,s,ft,n} = ff_{ft}(PR_u, PR_s, n) = \\ \{u' \in U | \, \forall \, X \subseteq (U \setminus \{u'\}) : \\ |X| < n \quad \vee \quad \exists \, x \in X : pred_{u,s,ft,u'} > pred_{u,s,ft,x}\} \end{aligned} \tag{2.8}$$

In case of large supplier profiles, a model-based filtering algorithm may be used instead, leading to results that may differ somewhat from this definition.

Table 2.2: Overview of Information Filtering systems and the results provided.

	supplier: $s = p$	supplier: $s = u''$
	Recommender System/ Hybrid IF System	distributed Hybrid IF System
input: i	$pred_{u,p,ft,i}$	$pred_{u,u'',ft,i}$
input: n	$REC_{u,p,ft,n}$	$REC_{u,u'',ft,n}$
	Matchmaker System	distributed Matchmaker System
input: u'	$pred_{u,p,ft,u'}$	$pred_{u,u'',ft,u'}$
input: n	$SU_{u,p,ft,n}$	$SU_{u,u'',ft,n}$

Table 2.2 summarizes the different kinds of IF systems and the results they provide.

We note that Matchmaker Systems and Hybrid IF Systems require a collaboration-based approach because they ultimately have to be based either on a global provider profile containing information about users, or on user profiles containing information about other users. Otherwise, a large fraction of all user-user pairs would have to be compared in order to find users that are actually similar, which is infeasible in real-world applications containing a large number of users. Feature-based approaches are only suitable for pure Recommender Systems, which by definition cannot be based on collaborative approaches.

Finally, all four kinds of results may be produced based on the part of a supplier profile returned as a result of a query on that profile. In this case, which constitutes the combination of long-term and short-term information needs described above, the given equations have to be modified by substituting $q_{IR}(PR_s)$ for PR_s. We refer to this case as the mixed IR/IF scenario.

Queries on a profile may also occur during the actual filtering process. We distinguish between both types of queries by denoting the former queries as $q_{IR}(PR)$ and the results as *constrained profiles*, and the latter queries as $q_{IF}(PR)$ and the results as *partial profiles*.

The main components of an IF architecture, including profiles and modules for the three stages described above, are shown in Figure 2.3 for the Recommender System scenario, i.e. with a provider as the supplier in the information filtering stage. The Matchmaker System scenario is largely analogous, with a second user replacing the provider.

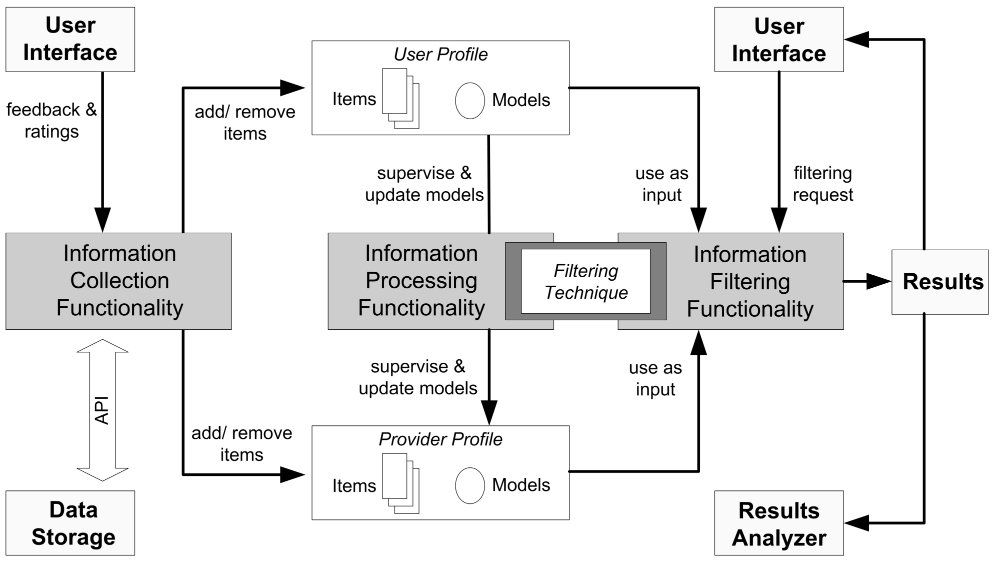


Figure 2.3: The main components of an IF architecture. The provider-side components (Data Storage and Results Analyzer) are used as placeholders for provider functionality that is not further specified here.

The two main groups of filtering techniques are classified further mainly by the kind of profile data they are based on. There are three feature-based approaches (and various combinations thereof):

- In *Content-based Filtering* approaches, results are generated by determining provider profile items that are similar to user profile items. Similarity is determined either by comparing attributes of the respective items directly, or by creating and applying models that are based on the attributes of items.

- In *Knowledge-based Filtering* approaches, results are generated by applying domain-specific knowledge to the user profile items and determining relevant provider profile items based on this knowledge.
- In *Utility-based Filtering* approaches, results are generated by applying a utility function, which describes a user's general preferences, to the provider profile items.

Additionally, there are two main collaboration-based approaches:

- In *Collaborative Filtering* approaches, similar users are identified based on the rated items and general preferences of the respective user profiles. If recommendations are generated, they are based on the profile items of similar users.
- In *Demographic Filtering* approaches, similar users are identified based on the demographic data of the respective user profiles. Personal preferences may additionally used to some extent in order to supplement or infer demographic data. If recommendations are generated, they are based on the profile items of similar users.

Combinations of pure feature-based approaches and collaboration-based approaches are classified as collaboration-based approaches as well, according to the definitions. The various approaches and combinations are described in detail in [19].

2.2.2 IF Architectures

As defined above, IF architectures contain three abstract entities aggregating various components, and operate on various data (such as profiles and results). Architectures differ, however, in the way the components and data are controlled, and in the actual entities aggregating the abstract entities. Existing IF architectures can be grouped into the following three main categories:

- In *provider-controlled IF*, the provider entity controls all components and data including the user profile data, and aggregates the filter entity as well. This architecture is the most common approach in research prototypes as well as E-Commerce applications of Recommender Systems. Usually, all data is stored in a central database. This approach allows personalized information to be generated in a highly efficient manner, because all required information can be accessed directly and in a uniform way. At the same time, as a centralized architecture constituting a single point of failure e.g. with regard to access to the user profiles, it is characterized by inherent security risks. Figure 2.4 shows this architecture.

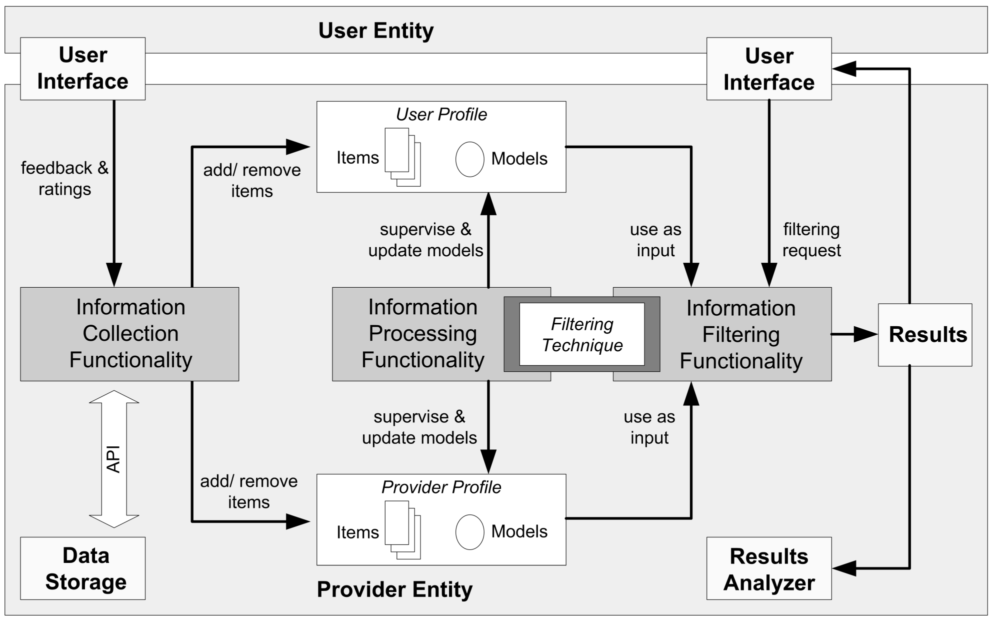


Figure 2.4: A provider-controlled IF architecture.

- In *privacy-enhanced IF*, the privacy of the user is addressed by enabling the user entity to store the user profile data locally, e.g. within a web browser plug-in or as part of a personal agent. Apart from this modification, the provider-controlled IF architecture model is used. This approach allows the user entity to determine the personal information that is collected. Figure 2.5 shows this architecture.

- In *user-controlled IF*, the user entity not only controls the user profile, but aggregates the filter entity as well. Thus, the architecture basically resembles the privacy-enhanced approach, with the status of user and provider reversed: In order to generate results, the user entity uses data from the provider profile. This approach is primarily used in collaboration-based approaches that are largely independent of a specific information provider. In this case a second user entity replaces the provider entity as supplier of the additional data. Compared to provider-centric approaches, however, its complexity is generally rather high: Additional infrastructure aspects have to be addressed

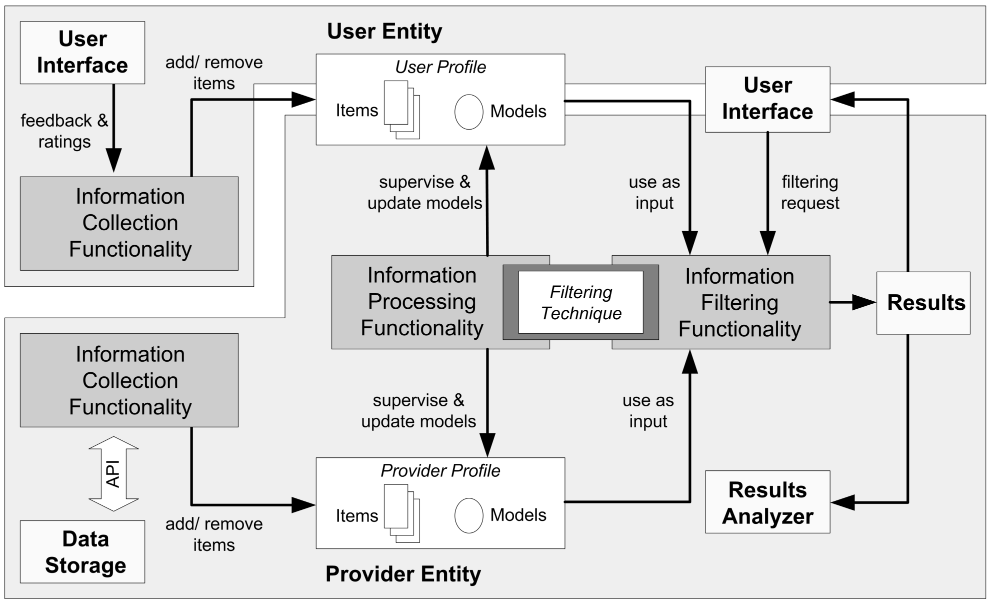


Figure 2.5: A privacy-enhanced IF architecture.

because more complex software on the user side is required, as well as an advanced communication network (such as a peer-to-peer network). Figure 2.6 shows this architecture.

2.2.3 Main Problems

IF systems have not gained widespread acceptance. There are various Recommender Systems and Matchmaker Systems available, usually as part of e-commerce applications[3], but they are generally limited to specific domains in which privacy and quality aspects in particular are less relevant, such as entertainment as opposed to health or finance, and they often constitute an additional service rather than core functionality. We identify the following four problems as the main limitations resulting in the lack of acceptance of IF systems:

- *Privacy*: Personalization and privacy are often seen as contrary and irreconcilable: Regardless of the chosen approach, personal information is always required as a basis for personalized information. Many users,

[3]Popular examples include online shops such as **amazon.com**, and internet radio stations such as **last.fm**.

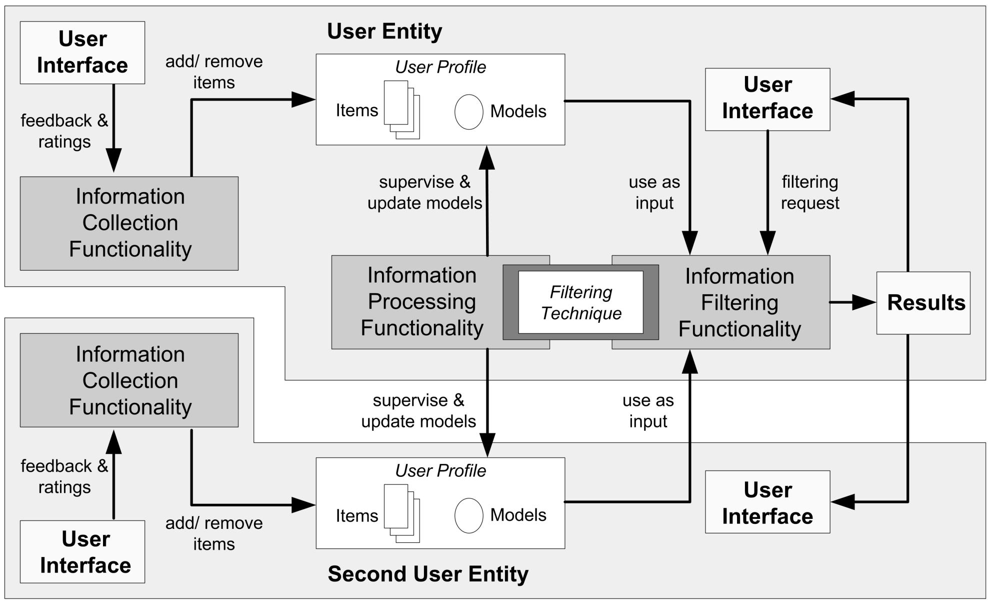


Figure 2.6: A user-controlled distributed IF architecture.

however, are reluctant to provide this information. None of the architectures described above offer a solution for multilateral privacy, i.e. a solution in which the privacy of all participating entities is protected:

– The main drawback of provider-controlled IF is the lack of user privacy: The user entity cannot determine the personal information that is collected. Furthermore, even if the provider entity ensures and actually attempts to protect the privacy of the user information[4], this cannot be verified or enforced by the user entity. Therefore, the user entity cannot prevent the propagation or, more generally, the unintended use of private data.

– The privacy-enhanced IF approach allows the user entity to initially control the propagation of his personal information by deciding which entities to grant access to this data. However, as the provider entity ultimately still has to be allowed to access the user profile in order to provide personalized results, the user entity still cannot prevent the propagation and unintended use of private data.

– The user-controlled IF approach is problematic with regard to provider privacy, if the provider entity participates actively in the

[4]Approaches for protecting the privacy of user data on the provider side are discussed in Section 3.1.4.

actual IF processes, because in this case the provider entity loses control over its profile. In the case of a distributed system, user privacy is still not protected completely because private data is exchanged between users.

The privacy of the filter entity, who may regard the details of the algorithms used by the filtering techniques as private data, is not protected by any of the approaches described here. We consider the lack of privacy to be the most important problem in IF architectures, and therefore it is the primary focus of this work. It is discussed in detail in the following section.

- *Quality*: Recommender Systems often return unsatisfying results because they are largely based on unfounded assumptions about the user and his information needs. In particular, the metrics used to determine which items to recommend to a user are inadequate in many cases [82], because they lead to returning obvious recommendations that are very similar to user profile elements, and neglect the aspect of serendipitous (unexpected) recommendations. Basically, metrics often tend to ignore the fact that the user may not always be interested in the elements with the highest prediction [122].

- *User Effort*: Users are often not willing to spend any amount of time learning to interact with unknown software or interfaces. Instead, they prefer to obtain results of lesser quality immediately, even if they would profit more from the former course of action in the long term. This behavior, the *Production Paradox*, is a significant aspect of the *Paradox of the Active User* as defined in [26]. In the context of Recommender Systems, this behavior is encountered in the fact that users are often not willing to explicitly provide the information their profile is based upon, e.g. by rating recommendations. Thus, Recommender Systems that depend largely on explicit user feedback discourage many potential users.

- *Provider Bias*: Commercial IF architectures ultimately focus on realizing the provider's goals, which may be contrary to the users' goals: While providers may offer personalized information with the ulterior motives of customer retention (a user may use the respective service in the future because he already put a certain amount of effort into creating his user profile) and customer data aggregation, these aspects have generally negative implications for users. The notion expressed by the term "Customer Relationship Management" is challenged by the fact that customers generally reject the idea of relationships with corporations.

To summarize, we identify the following main reasons for a lack of acceptance of existing IF-based systems: Lack of privacy, lack of quality, required user effort, and provider bias. Based on these problems, we arrive at requirements of a PPIF architecture in Section 2.3.4. Other problems in IF systems, such as security aspects, obviously have to be addressed as well, but are less specific to IF systems and therefore not discussed here.

2.3 Privacy & Information Filtering

In this section, we discuss the aspect of privacy in the context of IF architectures. In the following, we show that privacy is in fact the main problem in IF architectures, we introduce the concept of multilateral privacy, and discuss different aspects of privacy that have to be considered when designing an architecture for PPIF. Finally, we state the requirements for Privacy-Preserving Information Filtering.

2.3.1 Privacy as the Main Problem

Privacy is often stated to be the primary problem in IF architectures [24, 77]. This statement is, however, somewhat at odds with the actual behavior of a large number of users who can be classified as *privacy pragmatists* willing to waive privacy in return for other benefits or incentives [107]. Only a small number of users are *privacy fundamentalists* aiming at keeping all personal information private, regardless of its actual sensitivity. A comprehensive overview of surveys dealing with users' privacy concerns in the context of personalization is given in [108]. It observes widespread differences between stated preferences and the actual behavior of users. Nevertheless, the conflict of privacy and personalization is shown to be a significant problem in personalized electronic commerce applications, and the need for solutions is stressed.

The currently observed user behavior, however, has to be assessed in the context of existing IF architectures, which do not realize their full potential, mainly because they are restricted to the specific domains, as the following examples indicate.

- **E-commerce**: most current Recommender Systems assist a potential buyer only by recommending products of a relatively low value, such as books as opposed to cars, houses, or stock.

- **Lifestyle**: Current Recommender Systems are largely restricted to entertainment-related domains such as movies and music, ignoring e.g. healthcare and wellness-related products.

- **Knowledge**: Current Recommender Systems offer personalized newsletters for politics, local news, or sports, but they generally disregard the areas of academic publications, as well as financial and economic information.

In future IF systems offering these kinds of information, perhaps even in an integrated manner, users can be expected to be much more concerned about their privacy, basically because of the large amount and sensitive character of the personal information required in these cases, as opposed to current solutions.

2.3.2 Multilateral Privacy

In IF architectures which aim at becoming widely accepted, privacy aspects related to all main entities should be addressed:

- Protecting the *user privacy* is the most obvious problem, because the goals of privacy protection and personalization are inherently conflicting: By definition, users have to provide personal information in order to obtain personalized information.

- Protecting the *provider privacy* is a problem that is sometimes neglected, especially if the term "privacy" is only applied to individuals. Obviously, the provider information cannot be protected completely, at least in Recommender Systems, because recommendations are based on provider information. Nevertheless, an information provider is likely to be concerned about the dissemination of his information, which is his principal asset. It would be detrimental, for example, if competitors would be able to extract and re-create the entire provider information in a feasible way, e.g. via using artificial user profiles as means for retrieving a large number of recommendations.

- Protecting the *filter privacy* is a problem that is generally ignored in the literature. It seems reasonable, however, to regard the filter algorithms themselves as a valuable asset that should be protected, mainly because the quality of personalized information and thus probably the commercial viability of the overall system directly depends on the quality of the filtering techniques.

We therefore use the term “Multilateral Privacy”, analogous to the term 'Multilateral Security' introduced in [94], in association with a system in which the privacy of all participating entities is addressed, with no entity taking precedence over another.

2.3.3 Measuring Privacy

The requirement “privacy protection” as such is rather vague and cannot be measured objectively for a given IF system. Therefore, it has to be specified further, in particular with regard to the following aspects:

- Against what kind of adversary does private information have to be protected?
- What kind of threats have to be countered?
- To what extent has private information to be protected?

We discuss these aspects in the following sections.

2.3.3.1 Adversary Model

From the viewpoint of one entity or participant of an IF system, any other participant may be an *adversary* attempting to access and/or propagate private information in a way that has not been agreed upon. We distinguish between the following types of participants[5]:

- An *honest participant* carries out all actions exactly as specified and announced. Even in systems consisting entirely of honest participants, threats with regard to private information exist, as described below.
- An *honest-but-curious adversary* (sometimes referred to as “semi-honest”) carries out all actions as specified and announced, but tries to gain additional information by arbitrary means, e.g. by analyzing the exchanged information. As an example, an honest-but-curious provider entity in a IF system based on a centralized architecture returns recommendations as specified, but it may use the user profile data for additional purposes.

[5]The terminology has been introduced in [56] in the context of Secure Multi-Party Computation, for which see Section3.1.2, but it is applicable in this broader context as well.

- A *malicious adversary* tries to gain additional information by means that violate the protocols agreed upon. As an example, a malicious provider entity in a IF system based on a centralized architecture may not even return any result data, or he may deliberately return incorrect data in order to induce the user to provide additional private information.

It should be noted that an honest participant may be forced to deviate from the protocol agreed upon as a reaction to a detected threat. Following [121], we classify a role as an honest *defending party* in this case.

The honest-but-curious adversary model is obviously weaker than the malicious adversary model, and thus protecting private information against an honest-but-curious adversary is generally easier than protecting private information against a malicious adversary. If prevention fails, however, it may conversely be easier to at least detect a malicious action as a noticeable deviation from the protocol, while an honest-but-curious adversary may be harder to detect because the results are indistinguishable from the results provided by an honest participant.

Furthermore, while the honest-but-curious adversary model is in many cases sufficiently realistic because the adversary may be interested in correct results as well, or because a malicious adversary is detected at some point in time, in the context of Information Filtering both kinds of adversaries should be assumed. However, we consider it to be sufficient for participants to be able to detect malicious adversaries, instead of aiming at an architecture preventing them completely, because a probable risk of detection should deter malicious adversaries in most cases.

2.3.3.2 Threats

There are different threats with regard to private information that is propagated to a *recipient*, e.g. from the user entity to the provider entity. In [45], the following five main threats are identified:

- *Deception by the recipient*: The recipient of private information uses the information for other purposes, or purposes exceeding those stated or agreed upon.

- *Mission creep*: The recipient initially adheres to his stated policy with regard to private information, but expands the original purposes over time, without re-negotiating with or even notifying the respective party.

- *Accidental disclosure*: The recipient propagates private information accidentally, e.g. via discarded hardware.

- *Disclosure by malicious intent*: Private information is stolen by a third party with malicious intents.
- *Forced disclosure*: The recipient is legally forced to propagate private information, e.g. via a subpoena.

Not all of these threats are equally important in an IF architecture. In a Recommender System for restaurants, for example, the threat of forced disclosure seems somewhat far-fetched, considering the character of the private information.

It should be noted that with the exception of the first threat, all threats may emerge even in systems that initially contain only honest participants. Countering the first threat, which may originate from honest-but-curious as well as malicious adversaries, goes a long way towards countering the other threats. Therefore, an IF architecture realizing multilateral privacy should primarily focus on the threat of deception by the recipient.

2.3.3.3 Degrees of Privacy

As stated above, an all-or-nothing approach with regard to privacy is impractical in IF architectures, because a certain amount of information has to be propagated by definition. Therefore, we define an acceptable degree of privacy that privacy-preserving IF architectures should provide. This sufficient degree of privacy is characterized by the following aspects:

- *Computational vs. Information-Theoretic Privacy*: Information can be considered as private either if it is computationally infeasible for a second party to obtain it, or if it is theoretically impossible for a second party to obtain it. The first case, i.e. the case of computational privacy, relies on (unproven but widely accepted) intractability assumptions used e.g. as the foundation of an encryption scheme. The second case, i.e. the case of information-theoretic privacy, does not rely on these kinds of assumptions and therefore even prevents an adversary with unbounded computing power to obtain the information. While information-theoretic privacy is obviously stronger, it is generally impractical to achieve. We therefore consider computational privacy to be sufficient in the context of privacy-preserving IF architectures.
- *Aspects of Anonymity*: If an entity cannot be associated with its private information by a second party, it is considered to be anonymous in this regard. An infrastructure for anonymous communication supports anonymity but may not ensure it unconditionally, because it does not

address the problem that the communicated information itself may be sufficient for identifying an entity. Therefore, in IF architectures a user profile may be anonymous in the sense that the respective user entity is not directly identifiable, but the user's identity may still be deduced indirectly by combining profile elements. In some cases, even a small number of apparently uncritical profile elements may be combined successfully in order to identify a user, e.g. if the respective user profile can be determined to be the only one containing that specific combination of elements[6]. Therefore, anonymous communication in itself is not sufficient. Additionally, private information is exposed even when a user cannot be identified. This is problematic especially in cases where valuable, i.e. non-trivial and interesting information may be deduced from a specific combination of elements in a user profile (e.g. a successful stock portfolio or a list of related work pointing to scientific work in progress or a pending patent). These issues are discussed in detail in [77]. We conclude that in privacy-preserving IF it should neither be possible to associate user entities with profile items, nor a profile item with other profile items. Provider and filter entities, however, are not required to remain anonymous.

- *Unlinkability vs. Unobservability*: In the optimal case, private information is *unobservable*, i.e. an adversary cannot even determine whether the respective information exists at all. In IF architectures, user profile information is unobservable if another entity cannot determine whether a specific profile item is actually contained in any user profile at all. Private information is observable but *unlinkable* if an adversary is able to determine that the respective information exists, but cannot associate it with certainty with any participant or other information, i.e. if he cannot determine a link. According to the privacy spectrum defined in [95], unlinkability with regard to private information is further classified by the following categories:

  - *Beyond Suspicion*: From the adversary's point of view, all theoretically possible links have the same probability. Therefore, he cannot rationally suspect a specific participant or other information to be associated with the respective information.
  - *Probable Innocence*: From the adversary's point of view, there may exist links with a probability higher than that of other links. Still,

[6]As an example, a large majority of United States citizens may be identified correctly based on just three attributes: Zip code, birth date and gender [106].

for any given link, its probability is smaller than 0.5, i.e. it is more likely that the link does not actually exist.

  – *Possible Innocence*: From the adversary's point of view, a link exists with a probability $1 \geq p \geq 0.5$, i.e. the possibility that the link does not actually exist cannot be ruled out.

  Obviously, if private information can be associated with certainty with a specific participant or other information, unlinkability is no longer given and the information is *exposed*. All categories of the privacy spectrum are summarized in Table 2.3 with regard to user privacy in IF architectures. We consider the "probable innocence" degree of unlinkability to be sufficient for user privacy in the context of privacy-preserving IF architectures under the condition that the probability is close to $1/|U|$ for all theoretical links, i.e. the "beyond suspicion" degree is almost reached.

  Regarding provider privacy, all information that is not provided directly via recommendations should be unobservable. Regarding filter privacy, the filter algorithm, if regarded as one single element of information, should be unobservable as well: It is obviously observable that *some* filtering technique is applied, but it should not be observable that a *specific* filtering technique is applied.

- *Privacy of the Returned Information*: Finally, it has to be considered whether the returned information should be regarded as private as well, i.e. whether e.g. the provider entity should be allowed to obtain the recommendations generated for a given user entity. This course of action has the following advantages: The provider entity receives feedback about the information that it provides, which may be useful for improving the quality of the information, and for adding further information in areas that are highly demanded. Additionally, it may reduce the complexity of the underlying filtering technique algorithms, because the requirements with regard to privacy are somewhat relaxed. On the other hand, allowing the provider entity to obtain result data compromises the privacy of the user profile information at least indirectly, because the provider entity may attempt to infer user profile information via the recommendations. Determining an optimal trade-off between these aspects is problematic and depends on the given scenario. Therefore, we do not resolve this issue at this point and merely observe that a privacy-preserving IF architecture should preferably support both approaches and reach a decision in particular cases e.g. through negotiation between the participants. As a minimal requirement, result data should

Table 2.3: The categories of the privacy spectrum according to [95] with regard to user privacy in IF architectures. I' is the set of all observed elements, c is a constant. The given probabilities reflect the adversary's point of view.

	User-Element Association	Element-Element Association
Unobservability: Absolute Privacy	$I' = \emptyset$	$I' = \emptyset$
Unlinkability: Beyond Suspicion	$\forall u \in U$: $prob(i \in PR_u) = c$	$\forall i' \in I'$: $prob(\exists\, PR : \{i, i'\} \subseteq PR) = c$
Unlinkability: Probable Innocence	$\forall u \in U$: $prob(i \in PR_u) \leq 0.5$	$\forall i' \in I'$: $prob(\exists\, PR : \{i, i'\} \subseteq PR) \leq 0.5$
Unlinkability: Possible Innocence	$\forall u \in U$: $prob(i \in PR_u) < 1$	$\forall i' \in I'$: $prob(\exists\, PR : \{i, i'\} \subseteq PR) < 1$
Exposure	$\exists u \in U$: $prob(i \in PR_u) = 1$	$\exists i' \in I'$: $prob(\exists\, PR : \{i, i'\} \subseteq PR) = 1$

not compromise a user's anonymity. As a maximal requirement, a degree of unlinkability similar to the degree of unlinkability reached with regard to profile elements should be reached. Predictions are considered to be unproblematic as long as they cannot be linked to a specific user.

To summarize, a privacy-preserving IF architecture requires a degree of privacy characterized by the following aspects:

- computational privacy;
- unlinkability of private user information and the respective user, and of private user information elements among themselves;
- regarding user privacy, a degree of unlinkability assuring probable innocence with a sufficiently low probability threshold;
- regarding provider and filter privacy, unobservability (with the exception of returned information);
- (optionally) user privacy with regard to returned information.

2.3.4 Requirements

The functional requirements of a comprehensive Privacy-Preserving Information Filtering system are the same as in any other comprehensive IF system, and follow directly from the definitions given above:

- The system should provide sufficient functionality for realizing the three stages introduced in Section 2.2.1, namely the information collection stage, the information processing stage and the information filtering stage.

- The system should be able to return all different kinds of result data defined in Section 2.2.1, namely predictions of the relevance of specific items, top-n recommendations of items, predictions of the similarity of specific users, and top-n similar users for a given user. In other words, the system should be able to provide Recommender System functionality as well as Matchmaker System functionality. As non-distributed Matchmaker Systems and non-distributed Hybrid IF Systems would be difficult to realize in a privacy-preserving manner, we explicitly do not require the system to be able to provide the respective functionality, which is not strictly required anyway to cover all kinds of result data.

- Regarding filtering techniques, the system should be able to support feature-based approaches as well as collaborative approaches.

Additionally, we define several non-functional requirements of a Privacy-Preserving Information Filtering system. As described in Section 2.2.3, we identify the following main reasons for a lack of acceptance of existing IF-based systems: Lack of privacy, lack of quality, required user effort, and provider bias. The first two can be expressed directly as requirements. The issue of user effort is partially covered by introducing the requirement of broadness, i.e. by requiring a solution to be applicable in various domains and in combination with a wide range of filtering techniques, because in this case the user is able to re-use his personal profile and has to enter information only once for different providers. The aspect of provider bias is addressed indirectly by the other requirements, and by acceptance aspects discussed below. Finally, privacy-preserving IF systems must not disregard performance issues: While some trade-offs with regard to performance are to be expected and may be acceptable, the overall performance of the resulting systems should be comparable to the performance of centralized systems, because otherwise it would be infeasible to deploy the resulting systems in real-world scenarios, or they would not be accepted by users because of usability issues. Therefore, we also introduce the requirement of performance.

Security requirements such as the requirement of secure communication, or protection against denial-of-service attacks, are not listed explicitly because they are already covered, from the respective entity's point of view, by the requirements regarding privacy.

We define the requirements as follows:

- *User Privacy* (R_u): No linkable information about user profiles should be acquired permanently by any other entity or external party, including other user entities, apart from observations of single profile elements which can be linked to a specific user or to other observed profile elements with negligible probability. The adverb "permanently" is used here to specify that private information related to a specific user may be acquired by another entity temporarily as long as it is not propagated or processed further and as long as it is removed completely, resulting in a state identical to a hypothetical state in which it had never been acquired. Result data, i.e. recommendations, predictions, or similar users should be regarded as private in this sense as well, if possible. If the returned information is not private, user profile information that can be deduced directly from returned information may be propagated as well. User anonymity *per se* is not required but may be provided optionally. While these requirements are obviously somewhat weaker than theoretically possible from the user's point of view (the optimal protection is reached by requiring unobservability of user profile information), they constitute a realistic compromise because user privacy can still be considered to be protected adequately while the relaxed requirements enable feasible solutions in terms of complexity and quality, and allow the provider to obtain some feedback about the provided information.

- *Provider Privacy* (R_p): No information about provider profiles, with the exception of the returned information, should be acquired permanently by other entities or external parties at all, i.e. provider information remains unobservable. Additionally, the propagation of information is entirely under the control of the provider. Thus, it is ensured that the provider may prevent e.g. the automatic large-scale extraction of information.

- *Filter Privacy* (R_f): The algorithms used by the filter entity should not be acquired permanently by any other entity or external party. General information about the algorithm may be provided by the filter in order to allow other entities to reach a decision on whether to apply the respective filtering technique.

- *Quality* (R_{qq}): The quality of the returned information should be close to the level of quality achieved in traditional IF approaches (although a small decrease in quality may be acceptable as a trade-off).
- *Broadness* (R_{bb}): The architecture should not be restricted to single information domains, specific filtering techniques, or specific persistent storage mechanisms. It should be easily adaptable to various domains and environments in order to facilitate fast and efficient development.
- *Performance* (R_{pp}): The computational complexity and latency of the realized solution should be close to the performance achieved in traditional IF approaches (although a small decrease in performance may be acceptable as a trade-off).

In addition to non-functional requirements, we also introduce the following acceptance aspects:

- *User Acceptance* (A_u): A system meeting all requirements listed above may still not achieve a high degree of user acceptance, e.g. if users do not trust the underlying technology, or if the system lacks usability.
- *Provider Acceptance* (A_p): Furthermore, a system meeting all requirements listed above may not be commercially viable or valuable in any other way for the providers of the system and the filtering techniques, e.g. if no viable business model is found or if the costs of running the system are too high.

These acceptance aspects have to be kept in mind when addressing the requirements, because otherwise the resulting systems would only be of theoretical interest. Acceptance is generally achieved to a large degree by meeting the respective privacy requirement. Provider acceptance of user-controlled approaches is expected to be somewhat lower because these approaches may not allow the provider to obtain any information about users at all.

2.4 Multi-Agent Systems

Multi-Agent System technology are one possible choice for realizing a distributed PPIF architecture. Section 4.2.3 discusses this choice in more detail. In this section, we provide definitions of the main concepts of MAS technology, describe the basic functionality required for our approach, and discuss the problem of malicious hosts and possible solutions thereof.

2.4.1 Definitions

There is no universally accepted definition for the terms *agent* and *Multi-Agent System*. The IEEE Computer Society standards organization Foundation for Intelligent Physical Agents (FIPA) provides a set of specifications covering most aspects of MAS architectures (see [46, 47, 49, 48] for the most general specifications). Our approach is generally applicable to all architectures complying with these specifications and the following basic definitions:

- An *agent* is a software-based entity operating *autonomously*, *reactively*, and *pro-actively*. It has the ability to *interact* with other agents via a special Agent Communication Language (ACL). Autonomy here denotes the ability of an agents to carry out complex tasks independently, i.e. without the intervention of other agents or humans, while having control over its own actions and internal state, e.g. through running on its own internal execution thread. Reactivity denotes the ability of an agent to respond to changes detected in its environment. Proactiveness denotes the ability of an agent to act not only reactively, but also on its own initiative, e.g. triggered by its internal state. This definition closely follows [118]. An agent basically contains program code that enables it to operate according to this definition, knowledge, i.e. basically a set of data, and its internal state.

- An *agent service* is a specific task an agent may carry out internally or on behalf of another agent. In order to enable other agents to use its services, an agent usually has to announce these services by providing an abstract description in a well-defined format, i.e. based on an common *ontology*.

- An *agent platform* is the runtime environment of a group of agents. Agent platforms provide infrastructure functionality, such as yellow pages services for agent service discovery, and white pages services listing existing agents themselves. Additionally, agents on an agent platform are usually protected against threats originating from other agents on the same or any other platform, or from other external entities. The life cycle of an agent begins with the agent being created on a specific platform, and ends with the agent (or the entire platform he is located on) being terminated. Additionally, *mobile agents* have the ability to *migrate* from one platform to another, by transporting their program code, knowledge and internal state. An agent platform is provided by an *host* entity.

- A *Multi-Agent System* is the entirety of agents and agent platforms either deployed in the context of an application as a *distributed problem solving system*, or deployed with different and possibly conflicting goals in a more abstract common context as an *open system* [120].

2.4.2 Basic Functionality

Our approach requires a MAS with certain basic features and functionality which are largely implicit in the definitions given above: All interactions between agents are carried out via agent services. In order to exchange a message between a sender agent and a receiver agent, an agent service is used where the sender is the service user and the receiver is the service provider. The agent service is realized by both participants following a specific protocol, i.e. a user protocol on the service user side and a provider protocol on the service provider side. For basic interactions, the user protocol may be as simple as sending initial data and receiving the result data, but more complex user protocols may be required for complex interactions. Regarding unobservability of interactions, it should be possible to hide the content of an interaction from all possible observers, namely other agents, the agent platform itself as well as external entities. The fact that an interaction takes place may however be observable, e.g. by the agent platform. Agents should be able to control the access to the services they offer, which is done e.g. by using an Service Control List mechanism. An agent's program code and knowledge is considered private information that should not be accessible by other parties without the agent's consent. Regarding platform management functionality, we assume a special agent realizing a PlatformManagerRole that provides at least interactions for creating and terminating agents, and for migrating mobile agents. All agents realize a generic AgentRole which allows them to use services offered by agents realizing more specific roles.

2.4.3 Malicious Hosts

Features of agents such as autonomy and the capability to contain private information in the form of knowledge suggest the use of agents as *personal agents*, i.e. agents acting on behalf of a human user (or, more generally, abstract entity) and containing private information of the respective user or entity. In this context, the security risks of agents operating on a platform provided by a potentially untrustworthy host have to be addressed: While the protection of hosts against agents, and of agents against other agents in the same environment are relatively straightforward issues that may be addressed adequately (see e.g. [99]), the problem of protecting agents against malicious hosts is more complicated (see [111] for a survey). In addition to threats not necessarily originating from the host, such as threats related to communication of mobile agents, there are two main threats related to the following aspects of an agent running on a remote platform:

- *Privacy*: The host may attempt to obtain the program code and/or knowledge of an agent without interfering with the execution of the agent. In this case, the host is regarded as honest-but-curious with respect to the general threat model.
- *Integrity*: The host may attempt to tamper with the agent code and/or data during the execution of the agent. In this case, the host is regarded as malicious with respect to the general threat model.

In the context of personal agents, these threats directly affect the privacy of the user or entity represented by the personal agent. Apart from relying on trusted hosts, several approaches have been suggested to counter these threats, which are discussed in Section 3.3.2.

2.5 Summary

This chapter provides definitions that are used throughout this work. Based on these definition, the main problems of IF systems in general and the problem of privacy in IF systems in particular are described, and requirements for a Privacy-Preserving Information Filtering architecture are derived from this problem description.

Regarding definitions, this work focuses on informational privacy as one of several aspects of privacy (Section 2.1.1), and focuses on privacy protection via technology as one of several strategies for protecting privacy (Section 2.1.2). It deals with IF systems in the form of Recommender Systems, Matchmaker Systems, and Hybrid IF Systems, which provide functionality for three separate stages, namely the information collection stage, the information processing stage, and the information filtering stage. In the latter two stages, filtering techniques are used that are group into feature-based approaches and collaboration-based approaches (Section 2.2.1).

As part of the problem description, we describe different existing types of IF architectures (Section 2.2.2), and list the main problems of IF system (Section 2.2.3), out of which we highlight privacy as the main problem (Section 2.3.1).

We introduce the concept of multilateral privacy (Section 2.3.2), which we specify further in the context of IF systems by defining realistic adversary models (Section 2.3.3.1), by describing various threats to privacy (Section 2.3.3.2), and by defining a degree of privacy that reflects the interests of the involved entities adequately and at the same time is realistically achievable (Section 2.3.3.3). Subsequently, we list all requirements of a Privacy-Preserving Information Filtering architecture (Section 2.3.4).

As our solution is based on Multi-Agent System technology, we define the main concepts of MAS technology (Section 2.4.1); we list basic functionality required for our approach (Section 2.4.2); and we describe the problem of malicious hosts in MAS systems as the central problem related to privacy in a MAS context (Section 2.4.3).

Table 2.4: An overview of existing IF architectures in relation to the requirements and acceptance aspects of Privacy-Preserving Information Filtering. A requirement is fully met (indicated by "✓"), partially met (indicated by "○"), or not met at all (indicated by "–"). Acceptance is indicated in an analogous manner. Note that the ratings do not always indicate the best value theoretically possible for the respective architecture, but an average value.

	Privacy Requirements			Other Requirements			Acceptance	
	R_u	R_p	R_f	R_{qq}	R_{bb}	R_{pp}	A_u	A_p
provider-controlled IF	–	✓	–	○	✓	✓	–	✓
privacy-enhanced IF	○	✓	–	○	○	○	○	✓
user-controlled IF (collaboration-based)	○	✓	–	○	○	–	○	○

Table 2.4 summarizes the problems of existing IF architectures by listing the existing approaches for provider-controlled, privacy-enhanced and user-controlled IF in relation to the coverage of the requirements and acceptance aspects. In the following chapter, work related to the area of Privacy-Preserving Information Filtering is discussed based on the definitions provided in this chapter, and existing approaches are evaluated in the same manner.

Chapter 3

Related Work

In this chapter, we review the state of the art in privacy-preserving Information Filtering, including related approaches and building blocks thereof. Work related to specific areas of our approach that is not directly relevant for the overall approach, such as specific algorithms for filtering techniques, is discussed in the respective chapters.

The chapter is structured as follows: The first section discusses Privacy-Enhancing Technologies (PETs) as the fundamental building blocks of privacy-preserving architectures, including basic concepts and functionality used by these PETs. Section 3.2 discusses Privacy-Preserving Technologies (PPTs) as privacy-preserving approaches for related areas, such as Private Information Retrieval and privacy-preserving data mining, as well as privacy-preserving Information Filtering architectures, i.e. approaches comparable to our architecture for Privacy-Preserving Information Filtering. Each section contains evaluation of the respective areas in the context of PPIF. Figure 3.1 gives an overview of the different groups of related work and their relationships, i.e. the ways they build upon each other. These two sections comprise the state of the art of privacy-preserving Information Filtering.

Section 3.3 discusses related work in the area of Multi-Agent System (MAS) technology that is directly relevant for our solution, i.e. related work dealing with anonymous communication and the problem of malicious hosts. Section 3.4 summarizes the chapter by giving an overview of the applicability of the different approaches with regard to Privacy-Preserving Information Filtering.

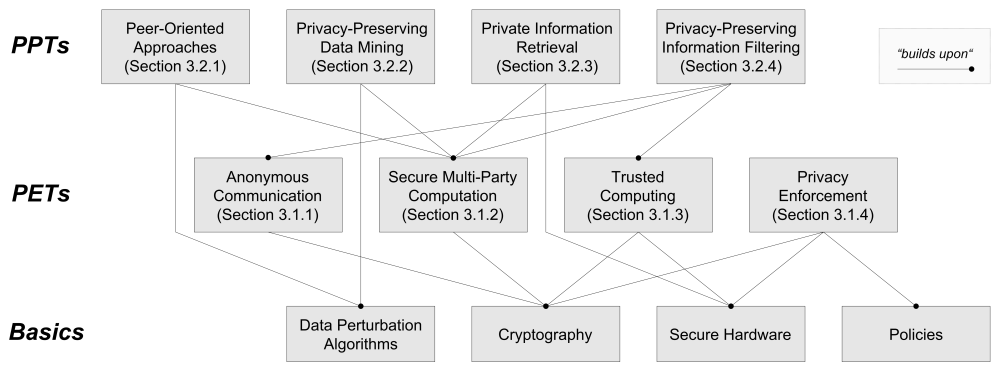

PPTs
Peer-Oriented Approaches (Section 3.2.1)
Privacy-Preserving Data Mining (Section 3.2.2)
Private Information Retrieval (Section 3.2.3)
Privacy-Preserving Information Filtering (Section 3.2.4)
"builds upon"
PETs
Anonymous Communication (Section 3.1.1)
Secure Multi-Party Computation (Section 3.1.2)
Trusted Computing (Section 3.1.3)
Privacy Enforcement (Section 3.1.4)
Basics
Data Perturbation Algorithms
Cryptography
Secure Hardware
Policies

Figure 3.1: Areas of research in PETs and PPTs. As indicated by the lines between areas, PPTs build upon the functionality provided by PETs, which in turn use basic concepts and functionality.

3.1 Privacy-Enhancing Technologies

Privacy-Enhancing Technologies have been researched under this designation for more than a decade. It should be noted that the designation is not used in all related work: There is a large amount of related work e.g. in the areas of anonymous communication and secure multi-party computing that is not explicitly labeled as PETs. Surveys of the field are given e.g. by [55] and [54].

In the following, we distinguish between PETs and PPTs by defining Privacy-Enhancing Technologies as basic building blocks which, while they may be used directly for a specific purpose, such as anonymous communication, have to be combined with other functionality in order to realize Privacy-Preserving Technologies used within applications for broader purposes, such as PPIF. In other words, in a complex scenario a single Privacy-Enhancing Technology (PET) may be used to enhance privacy, as the name implies, but is not sufficient for preserving privacy under all circumstances.

In addition to work on the theoretical foundations, research in the area currently focuses on four main areas:

- Anonymous communication over the internet and in peer-to-peer networks, including work on traffic analysis and related issues, such as anonymous web browsing, is discussed in Section 3.1.1.

- Protocols for Secure Multi-Party Computation are discussed in Section 3.1.2.

- Trusted Computing mechanisms and applications are discussed in Section 3.1.3.
- Other provider-side technologies for privacy enforcement, such as enterprise privacy policies and DRM-related approaches, are discussed in Section 3.1.4.

Research on privacy-enhanced applications, e.g. in the areas of face recognition, public transport, ubiquitous computing, and e-learning is not directly relevant in the context of Privacy-Preserving Information Filtering and therefore not discussed further here.

3.1.1 Anonymous Communication

With regard to user acceptance and prevalence, PETs for anonymous communication are arguably the most successful group of PETs to date. They are usually realized in the form of tools for anonymous e-mail and browsing, but may address related issues, such as anonymous communication in peer-to-peer networks as well.

The following terminology is adapted from [91][1]: A *sender* sends *messages* to a single *recipient* or a group of recipients via a communication network. An *attacker* is interested in obtaining information about these messages, e.g. about communication patterns, and may attempt to manipulate the communication. The content of the messages themselves is usually not considered in this context, because it is expected to be encrypted and therefore protected against attacks. In this aspect, the concepts discussed in the following have to be distinguished from the concepts introduced in Section 2.3.3.3, where we focus on the actual information, i.e. the content of the messages. However, as indicated by the use of similar terminology, such as "unobservability" and "unlinkability", in both contexts, the respective concepts are largely analogous. In the context of anonymous communication, we distinguish between the following concepts:

- *Anonymity* is a feature of an entity (a sender or recipient) who is not identifiable within a set of entities of the same type, the *anonymity set.*
- *Unlinkability* is a feature of a number of entities or messages who appear no more related after an attacker's observation than they are based on his prior knowledge. Anonymity may therefore be defined in terms of unlinkability:

[1]An extended version of this work is available online as version v0.31 via the URL <http://dud.inf.tu-dresden.de/literatur/Anon%5FTerminology%5Fv0.31.pdf>.

- *Sender Anonymity* is established in systems where, out of the sets of all senders and messages, there exists no sender and message that can be linked.
- *Recipient Anonymity* is defined in an analogous manner for recipients and messages.
- *Relationship Anonymity* is defined in an analogous manner for senders and recipients. It is a weaker notion of anonymity because it is implied by sender anonymity as well as by recipient anonymity.

• *Unobservability* is a feature of entities who cannot be distinguished from any other entity of the same type. Analogous to the definitions above, *Sender Unobservability*, *Recipient Unobservability*, and *Relationship Unobservability* are defined. It should be noted that unobservability always implies anonymity. Sometimes the term *Untraceability* is used synonymously.

• *Pseudonymity* is a feature of entities who use one or more pseudonyms for identification.

In the following, we review the main groups of solutions, which mainly focus on achieving sender and/or relationship anonymity. They may be extended by various methods (such as dummy traffic, or the use of steganography) to achieve a certain degree of unobservability. Exemplary approaches for recipient anonymity are broadcast mechanisms and Private Information Retrieval schemes, for which see Section 3.2.3.

3.1.1.1 Mix networks

The most basic approach for anonymity through unlinkability is the use of relays, also known as *proxies*, for all communication in order to hide the actual sender and/or recipients. Simple proxies, however, are vulnerable to traffic analysis threats. Mix networks (first suggested by [28]) address this problem by providing a number of proxies as *mixes*. Each messages is routed through various mixes, and each mix withholds messages until a certain amount of messages have been collected, and then re-encrypts and propagates messages in permutated order.

Subsequent Research on mix networks has generally segmented into two groups of approaches: High-latency approaches, such as the Mixminion approach [36], aim to minimize breaches of anonymity by introducing large and variable latencies, and are therefore less suitable for immediate communication, such as web browsing. On the other hand, low-latency approaches, such as onion routing [57] and its more recent modifications [39], aim at anonymizing network traffic itself.

While mix networks have been studied extensively (see e.g. [39] for an overview), they have been observed [23] to largely rely on insufficient cryptographic constructions, which has led to a large number of schemes that have been broken and modified, sometimes repeatedly. Nevertheless, existing applications for anonymous communication are mainly based on mix networks.

3.1.1.2 DC-Nets

While mix networks may be used to provide computational privacy (as defined in Section 2.3.3.3), DC-Nets are an alternate approach providing information-theoretic privacy as well as sender unobservability. DC-Nets have been introduced in [27]. The name is derived from the so-called *dining cryptographers problem*, which illustrates anonymous communication via a simple scenario: Three or more entities intend to receive a message (in the most basic case with a length of one bit) sent by one of the entities who intends to remain anonymous. The entities arrange themselves as a ring, i.e. a circular linked list, and each entity shares a secret bit with each of his two neighbors. Starting with a designated entity and a result bit set to zero, each entity in turn combines the two secret bits obtained from its neighbors, the message itself (in case the entity is the sender), and the result bit via the XOR-operation. Once this operation has been carried out by all entities, the result bit represents the message, without any information about the identity of the sender having been revealed.

In more general terms, the protocol realizes a superimposed sending of a message on a ring network, with each message bit requiring one round of communication around the ring for superimposing the message, and one round for broadcasting the result. Obviously, all entities are required to act in a non-malicious way, as a malicious entity may easily alter the message or disrupt the protocol. The main drawback of DC-Nets is the communication complexity which makes their use infeasible for real-world multi-user systems.

3.1.2 Secure Multi-Party Computation

Secure Multi-Party Computation (SMPC) protocols are the main building block for interactions involving two or more entities who intend to exchange some information while keeping related information private. The concept has been introduced in [119] as secure two-party computation and later generalized in e.g. [56]. An SMPC protocol involves n entities with inputs $I_1 \dots I_n$ and a function $f = (f_1, .., f_n)$ for computing outputs $O_1 \dots O_n$, based on these inputs, i.e. a function denoted by

$$(I_1, .., I_n) \mapsto (O_1 = f_1(I_1, .., I_n), .., O_n = f_n(I_1, .., I_n)).$$

Privacy is preserved by the protocol if the information an entity obtains during the protocol does not exceed the information the respective entity could derive from its own input and from the output of the protocol.

In this definition, all information is regarded as sensitive with regard to privacy. If additional information can be obtained, the respective protocol is therefore not privacy-preserving in a strict sense. It may still be sufficient, however, if the disclosed information does not actually violate the privacy of a participating entity. While the respective protocol may be optimized in terms of complexity by allowing additional information to be disclosed, it is generally more difficult to prove its sufficiency with regard to privacy.

Generic SMPC protocols model the function to be computed as a combinatorial circuit, and carry out comparatively small sub-protocols for every gate of the circuit. While theoretically applicable to a large class of functions, generic protocols are often practically infeasible because of large input sizes and the fact that complex functions have to be modeled as complex circuits, leading to inefficient protocols. Nevertheless, applications for realizing generic Secure Multi-Party Computation protocols have been introduced [81] and may be used for simple functions.

The following examples for simple functions are given in [34] in the context of Privacy-Preserving Data Mining, which is discussed in Section 3.2.2:

- *Secure Sum*: For values $a_1 \dots a_n$ distributed between multiple participants, the sum of the values is basically computed in the following way if it is to be known to lie in the range $[0..n]$: A designated first participant adds a random number to his local value and propagates the result, among all participants, with each participant adding his own local value. In each step, n is subtracted from the result if it is greater than n. Finally, the first participant subtracts the random value and thus obtains the actual result.

- *Secure Set Union*: For sets $A_1 \dots A_n$ distributed between multiple participants, the union $A_1 \cup .. \cup A_n$ is computed by each participant encrypting, via a commutative encryption scheme[2] , all items of his own set, and the encrypted items of all other sets, received from the other participants. At the end of the process, each item has been encrypted

[2]In commutative encryption schemes, the results is not affected by the order in which encryption or decryption function are applied, i.e. $\{\{m\}_{K1}\}_{K2} = \{\{m\}_{K2}\}_{K1}$.

n times, and because of the commutativity of encryption equal result values imply equal items. Thus, duplicates may be removed, and the resulting union set of encrypted items is decrypted by each participant in turn, resulting in the union set of decrypted items.

- *Secure Size of Set Intersection*: For sets $A_1 \dots A_n$ distributed between multiple participants, the size of the intersection set $A_1 \cap .. \cap A_n$ is computed by each participant encrypting, via a commutative encryption scheme, all items of his own set, and the encrypted items of all other sets, received from the other participants. At the end of the process, each item has been encrypted n times, and because of the commutativity of encryption equal result values imply equal items. Therefore, the size of the intersection set may be determined by each participant simply by counting the number of encrypted items appearing in each of the sets.

- *Scalar Product*: Apart from general approaches based on secure multiparty computation, which are not efficient, the problem of determining the scalar product of vectors in a privacy-preserving way has only been addressed for two parties, with various complex solutions e.g. based on the use of random vectors and matrices.

Private Information Retrieval schemes, described in Section 3.2.3, may also be based on SMPC protocols, because they basically realize the function

$$(i, \{x_1, .., x_n\}) \mapsto (x_i, \lambda),$$

i.e. the first entity retrieves x_i for a given i, while the second entity learns nothing at all.

3.1.3 Trusted Computing

Trusted computing aims at realizing trusted systems by increasing the security of open systems (i.e. systems which are accessed by various groups of entities, where entities of one group do not necessarily trust entities of other groups) to a level comparable with the level of security that is possible in closed systems (i.e. systems which are accessed by a single group of entities that trust each other). It is based on a combination of tamper-proof hardware and various software components. A trusted computing architecture has to address two aspects: *Integrity* of the system, which is achieved by ensuring the system cannot be tampered with in any way, and *authenticity* of the system, which is achieved by ensuring that a remote party can be convinced of the integrity of the system. The aspects are realized by three main mechanisms:

- *Secure Bootstrapping* ensures the system is initialized resulting in a state that adheres to a given security policy, e.g. by booting into a trusted operating system.
- *Strong Isolation* prevents the initialized system from being tampered with, and prevents applications from tampering each other.
- *Remote Attestation* certifies the integrity of software run on the system to a remote party.

A secure bootstrapping mechanism (described e.g. in [7]) basically enables a system to measure its own integrity during the boot process, and terminate the process if the integrity is compromised, via a chain of identity checks: A tamper-proof hardware device, the *trusted module*, starts the boot process by recomputing the hash of the BIOS, and compares this value with a hash of the BIOS signed with the private key of the trusted module and stored within it. The private key itself is embedded in the trusted module, and the respective public-private key-pair is certified by a Certificate Authority (CA)[3]. If the values match, the BIOS has not been tampered with, and control is passed to it. The process is continued by the BIOS and the next layer in the chain in a similar manner, and by subsequent layers up to the operating system. Thus, each layer certifies the next layer in the chain, by signing a hash of its executable image, and its public key. When the process is completed successfully, the system is guaranteed to have booted into a trusted operating system.

Secure bootstrapping alone does not allow a remote party to verify the integrity of the boot process. This is addressed either by extending the process to *authenticated boot*, which we do not describe here, or through remote attestation. In other words, secure bootstrapping restricts the software that may actually run on a system, while remote attestation reports which software runs on a system. It is up to the remote party to decide how to deal with the information given.

The basic mechanism of remote attestation, however, is similar to the secure bootstrapping mechanism described above: An application is attested by the operating system signing a hash of the executable of the application.

[3]It should be noted that for this reason, trusted computing requires a trusted third party essentially certifying that the trusted module works as specified. This holds even when a central CA is replaced by a more flexible mechanism enabling direct anonymous attestation [16].

This certificate is sent to the remote party, along with all other certificates of the chain starting at the trusted module and ending at the operating system. The remote party verifies each certificate (thus verifying the integrity of the boot process as well), and checks the corresponding hashes against a list of approved soft- and hardware. Remote attestation should result in a secret shared between the application and the remote party, otherwise it cannot be ensured that the attested application is actually executed. We discuss problems and their suggested solutions related to remote attestation in the following section.

Finally, strict isolation is handled by the trusted operating system. Virtual machine monitors may be used in order to abstract from the actual hardware, and leverage strict isolation by running applications in different virtual machine monitors [51].

3.1.3.1 Semantic Remote Attestation

According to [60], the basic remote attestation process has the following problems:

- *Program behavior is not attested*: The only information provided is that a certain executable is running. It is up to the remote party to determine whether this executable actually runs as specified, or it has to be trusted. Both alternatives are obviously problematic especially in cases where sensitive information is involved. Even if the executable is actually intended to act non-maliciously, it may fail to do so because of bugs or design flaws. In any case, it is impossible for the average human user to analyze an executable.

- *Inflexibility*: Remote attestation is carried out once, before the executable starts to run. Therefore, information about its runtime state or input data cannot be provided.

- *Management issues*: Updates or patches of executables result in different hashes. The remote party has to update its list of approved executables accordingly, again with the problem that program behavior is not attested. The problem is exacerbated by the fact that in many cases multiple patches and updates exist and are applied in various order, resulting in a large number of different executables. Additionally, the number of executables that have to be approved is increased by the fact that platform-specific binaries are attested, i.e. the same software on different operating systems results in different executables.

- *Revocation*: A problem inherited from public-key cryptography is the revocation of certificates, which cannot be addressed easily in an efficient way. To make sure a certificate is valid, Certificate Revocation Lists would have to be checked for every attestation process.

In semantic remote attestation [60], some of these problems are addressed by using language-based techniques in combination with a virtual machine approach, with the goal of attesting program behavior rather than particular executables. Instead of the trusted operating system attesting the executable of an application, the virtual machine, which is capable of executing platform-independent code, attests various properties of an application running within it. This is possible because in order to be executable within a virtual machine, the respective code contains high-level information which may be used for attestation.

3.1.3.2 Applications

Trusted computing is most often discussed in relation with Digital Rights Management (DRM), i.e. as a mechanism for realizing provider privacy by limiting users' access to information. There is, however, a large number of other potential applications, including mechanisms for realizing user privacy: Some example applications, including peer-to-peer networks, distributed firewalls, and distributed computing in general, are listed in [51]. Other obvious potential applications are MAS architectures supporting mobile agents, anonymous remailers, PPIF, and other PPTs.

In trusted computing, tamper-proof hardware is used only for the bootstrapping process. Related approaches use a secure coprocessor as tamper-proof hardware for additional tasks (see e.g. the Private Information Retrieval (PIR) schemes discussed in Section 3.2.3). This course of action focuses more on prevention of tampering, while trusted computing focuses on detection of tampering. Remote attestation, however, is often not addressed explicitly by these approaches, and they require customized tamper-proof hardware.

3.1.4 Privacy Enforcement

In this section, various approaches for privacy enforcement are discussed. They are based on the assumption that a given provider of information services intends to actually protect the privacy of his customers, i.e. the users, and thus does not act in a malicious manner. A basic requirement for privacy enforcement is the following: Privacy policies of providers as well as privacy preferences of users have to be expressed in a structured form, i.e. in a form that allows the user preferences to be processed and applied to the user profile data automatically, according to the respective privacy policy. The Platform for Privacy Preferences Project specification [90] meets this requirement by defining a syntax and semantics for privacy policies, as well as mechanisms for associating privacy policies with web resources. It does not offer support for actual provider-side enforcement of privacy policies.

A basic kind of enforcement make take place at the user side, through programs which for example fill out web-based forms according to the user's preferences and the respective web site's privacy policy. User-side enforcement can only be used to restrict the information a provider receives, but not to control further dissemination of this information. The latter aspect is addressed by provider-side enforcement, which aims at protecting sensitive data by various means, listed as follows:

- *Hippocratic Databases*: In analogy to the *hippocratic oath*, which protects the privacy of patients, a mechanism for protecting user data in databases is described in [2]. Basically, database records are extended by adding privacy metadata, derived from the information a user has specified in a Platform for Privacy Preferences Project (P3P) profile and the provider's privacy policy. Based on the privacy metadata, sophisticated access control mechanism are implemented. Additional tools are utilized to detect unusual and potentially privacy-critical queries, and to record an audit trail for each query.

- *Enterprise P3P*: A similar approach is described in [69], resulting in a *Platform for Enterprise Privacy Practices* in which every data object is enhanced by metadata information specifying the respective access policy. Thus, data may be handled according to a privacy policy and a user's preferences at all stages of an enterprise process, not only in the context of a database.

- *RAIC-based dynamic adaption*: A component-based approach is described in [72] resulting in an architecture based on a Redundant Array of Independent Components where a simple component provides functionality adapted to specific privacy constraints, e.g. a specific filtering technique in an IF context. Suitable components are selected for each process based on the current privacy constraints, such as the privacy preferences of the current user.

With regard to the privacy threats listed in Section 2.3.3.2, privacy enforcement primarily deals with the threat of accidental disclosure.

3.1.5 Evaluation

While Privacy-Enhancing Technologies are useful building blocks, each group of PETs described in this section is, in itself, not sufficient for realizing a privacy-preserving IF architecture in a feasible way, because an IF system based on a single group of PETs does not meet all requirements listed in Section 2.3.4:

- Anonymous communication between user and provider may not prevent the provider from identifying the user via the user's profile data, and it may allow the provider to obtain private information by aggregating profile data. The other non-functional requirements would not be affected decisively by the use of anonymous communication. Because of its prevalence as a PET, it is generally accepted by users. Providers, however, may not accept it as readily because they usually prefer to be able to identify the user they are interacting with.

- Secure multi-party computation protocols are infeasible for large quantities of data and complex functions, such as filtering technique algorithms. Additionally, they cannot be used if the function to be computed is to be kept private as well, which is required for filter privacy. These PETs is not likely to affect acceptance aspects decisively.

- Trusted computing by itself is not sufficient for realizing a generic privacy-preserving IF architecture, mainly because of practical issues related to remote attestation of an application meeting the requirement of broadness (these issues are discussed in detail in Section 10.2), and because of a lack of user acceptance with regard to trusted computing in general.

- Privacy enforcement mechanisms at the provider side do not protect sensitive user data against malicious providers. These PETs is not likely to affect the other requirements and acceptance aspects decisively.

Table 3.2 summarizes the evaluation of all PETs in the context of all work related to PPIF. To recapitulate, functionality from different groups of PETs has to be combined with additional functionality in order to realize the goal of Privacy-Preserving Information Filtering.

3.2 Privacy-Preserving Technologies

In this section, we discuss Privacy-Preserving Technologies as work that is related to our approach. While these PPTs are not directly applicable for

PPIF, mainly because they are intended for different purposes, they are nevertheless relevant because the problems and concepts introduced are similar to those of PPIF. Furthermore, we discuss related work in privacy-preserving IF architectures.

3.2.1 Peer-Oriented Approaches

Peer-oriented approaches are protocols based on Secure Multi-Party Computation that allow a group of similar entities, i.e. entities characterized by having equivalent goals and uniform private data structures, to accomplish various tasks related to Information Retrieval and Information Filtering. Hence, these approaches are not applicable in scenarios containing entities with different goals (such as user and provider). In the following, we discuss two exemplary peer-oriented approaches. For other approaches, see e.g. [22].

3.2.1.1 Privacy-Preserving Indexing

In this scenario, described e.g. in [13], a document collection is shared among multiple entities, e.g. in a peer-to-peer file-sharing setting, and a global index is to be created in order to enable documents to be retrieved in a more efficient way than by asking each entity whether it may be able to provide the document in question. However, the participating entities do not wish their entire partial collections to be known globally, and they intend to be able to decide whether to provide a document, based on the entity requesting the document. The protocol suggested in [13] basically introduces a number of false positives for each document equal to the number of true positives (the entities actually sharing the respective document). The global index contains all false and true positives, and thus the probability that an entity listed in the index as sharing a specific document actually does so is 0.5. In terms of unlinkability (see Section 2.3.3.3), this solution therefore provides an unlinkability degree close to “probable innocence”. However, the privacy-preserving construction of the index turns out to be problematic in the case of colluding malicious entities.

3.2.1.2 Privacy-Preserving Clustering

Another peer-oriented approach closely related to Information Filtering is privacy-preserving clustering, in which two or more entities aim at constructing a global model of clusters of private data. Apart from relying on a trusted third party, there are two basic approaches:

- *Data Perturbation*: Before the clustering algorithm is applied, noise is added to the underlying data.

- *Secure Multi-Party Computation*: The clustering algorithm is based on a SMPC protocol.

Perturbation-based approaches are described e.g. in [84]. In [66], an SMPC-based approach for privacy-preserving clustering is described which is based on the k-means clustering algorithm for two parties under the assumption of a honest-but-curious adversary, which utilizes a privacy-preserving protocol for computing cluster means. Two approaches for the protocol itself are described: A protocol for oblivious polynomial evaluation [86], and a protocol for homomorphic encryption [14].

3.2.2 Privacy-Preserving Data Mining

Data mining, also known as Knowledge Discovery in Databases (KDD), deals with the acquisition of non-obvious, potentially useful information, via an automatic extraction of previously unknown patterns from large amounts of data usually stored within databases. This is generally achieved by building models aggregating the raw input data. The output generated by a data mining process is a classifier or rule describing the previously unknown pattern. In addition to applying standard machine-learning techniques to generate the output, algorithms based on association rule mining are often used (see [62] for a survey), producing association rules, which describe relations between sets of items expressing statements of the form "users who bought item x and y also bought item$z$".

Privacy-Preserving Data Mining aims at preserving the privacy of users providing the input data, basically via two distinct approaches:

- *Partial privacy*: The model on which data mining processes are to be applied may contain sensitive information, but this information is hidden in the output.

- *Complete privacy*: Sensitive information is hidden even in the model itself, though it may be contained within the input data.

Based on the classification given in [112], we distinguish between the following three classes of approaches for preserving privacy:

- *Heuristic-based approaches* are utilized to obtain partial privacy, i.e. they are applied when output is generated from a model.

- *Cryptography-based approaches* are utilized to obtain complete privacy. They are applied to create a model by secure multi-party computation, based on input that has not to be revealed itself.
- *Reconstruction-based approaches* are utilized to obtain complete privacy. They allow the perturbation of the input data, resulting in a model containing no sensitive information. Because this model is based on perturbed data, the actual model has to be reconstructed from it.

3.2.2.1 Heuristic-Based Approaches

Heuristic-based approaches aim at preventing certain patterns in the respective data set, such as rules based on sensitive information. In the case of association rule mining, the problem is formalized as follows: Given a set of rules R and a subset $R_h \subset R$, the underlying data set D has to be transformed to a data set D' in a way that allows the mining of rules R, but not of R_h. In other words, heuristic based approaches aim at limiting possibilities for data mining. For this reason, and because they only achieve partial privacy, they are not relevant as related work with regard to PPIF, and therefore not discussed further here.

3.2.2.2 Cryptography-Based Approaches

Cryptography-based approaches based on secure multi-party computation have first been suggested in [78]. Generic solutions are not applicable for privacy-preserving data mining in a feasible manner, because large data sets and complex algorithms are involved. Therefore, specific protocols for secure multi-party computation have to be designed.

In [78], a secure two-party computation protocol for the a decision tree learning algorithm is described realizing the function

$$(D_1, D_2) \mapsto (ID3(D_1 \cup D_2), ID3(D_1 \cup D_2)),$$

i.e. given two participants with separate databases D_1 and D_2, each participant obtains a decision tree based on the joint database. The communication complexity of this protocol is shown to be reasonably close to the communication complexity of a non-private protocol for distributed computation of the decision tree.

In [34], basic operations are described as a toolkit for realizing various efficient data mining algorithms, as listed in Section 3.1.2. Using a combination these basic operations, algorithms for association rule mining in horizontally partitioned data (transactions are distributed between participants) [68] and for association rule mining in vertically partitioned data (each transaction is distributed between participants) [110] are described. While the resulting data mining techniques are not secure multi-party computations in the strict sense, because intermediate information is disclosed in addition to the output, they are shown to still preserve privacy to a large extent, by ensuring controlled disclosure of information. In other words, the information disclosed in intermediate stages of the algorithm may be regarded as being not sensitive, e.g. because it may give only a very general idea about the underlying data.

3.2.2.3 Reconstruction-Based Approaches

The original privacy-preserving data mining approach described in [3] is a reconstruction-based approach using a value distortion mechanism in which input data attribute values are perturbed by adding a random value (based on a uniform or gaussian distribution). In the reconstructed model, the original data distribution is reconstructed with sufficient accuracy, i.e. the decision tree classifiers generated as output based on the reconstructed model are similarly accurate to classifiers generated based on a model created from non-perturbed data.

A related approach [44] deals with creating association rules based on a reconstructed model. Recommender systems are explicitly stated as a possible application. The association rules are created based on randomized transactions, i.e. transactions in which each original items is replaced with a certain probability. Again, the resulting association rules are sufficiently accurate to be used instead of rules based on non-perturbed transactions.

3.2.3 Private Information Retrieval

Private Information Retrieval deals with retrieving information via a query mechanism, usually from a database, without revealing information about the query and the retrieved results itself to the entity providing the information. It has been introduced in [30]. Formally, the problem is expressed as follows: Given a database D as a binary string x of length n, i.e. $D = x_1 \cdots x_n$, a user has an index i and wishes to obtain x_i, without the entity hosting the database obtaining i or x_i. In a more general definition, the database contains more complex database records with a fixed maximal length. Obviously, retrieving the entire database constitutes a trivial solution to this problem. This approach has communication complexity of $O(n)$ and is therefore infeasible for large databases. Additionally, it may violate the privacy of the information provider. In the following, we discuss various classes of schemes for PIR. Table 3.1 gives an overview of the discussed approaches.

Table 3.1: An overview of suggested Private Information Retrieval schemes. Each solution is shown with its communication complexity.

	information-theoretic privacy	computational privacy
k-server schemes ($O(n)$ computational complexity)	[30] $O(n^{1/3})$ [6] $O(n^{1/(2k-1}})$	[29] $O(n^{\varepsilon})$
single-server schemes ($O(n)$ computational complexity)	trivial solution $O(n)$	[75] $O(n^{\varepsilon} log\, n)$ [21] $O(log^{d} n)$
single-server hardware-based schemes (up to $O(1)$ on-line computational complexity)	n/a	[104] optimal [8] optimal

The first approaches to PIR are based on replicating the database by distributing copies of the original database on different servers:

- In [30], a two-server PIR is described that has a communication complexity of $O(n^{1/3})$.
- In [6], this solution is generalized to a k-server PIR scheme with communication complexity $O(n^{1/(2k-1}})$.
- In [29], a two-server solution with an improved communication complexity $O(n^{\varepsilon})$ for any $\varepsilon > 0$ is described. The improved complexity is reached by replacing the goal of information-theoretic privacy with the goal of computational privacy, as defined in Section 2.3.3.3.

In all multi-server approaches, the respective providers are assumed not to collude, acting in a non-honest manner. This requirement turns out to be rather unrealistic, mainly because it is difficult to enforce, even more so as the respective entities have to communicate and thus know each other in order to keep the databases consistent. Therefore, later approaches are based on single-server schemes and achieve computational privacy rather than information-theoretic privacy (which in single-server PIR schemes can only be reached via the trivial approach):

- In [75], the first single-server PIR scheme is described, relying on an intractability assumption. The communication complexity is $O(n^{\varepsilon} logn)$ for any $\varepsilon > 0$. Basically, the approach is based on encrypting the query and applying the encrypted query to the database in a way that the provider is not able to recognize either the query or the result.

- In [21], a single-server PIR scheme with polylogarithmic communication complexity $O(log^{d} n)$ with $d > 1$ is described, based on a different intractability assumption. This communication complexity is close to optimal, because retrieving a bit from a binary string in the non-private case has a communication complexity of $O(logn)$.

Single-server solutions, however, require a computational complexity of $O(n)$, because the server has to access all data in every single PIR process, because otherwise information could be obtained via observing which data is actually accessed. For this reason, the single-server solutions described above, while theoretically viable, are considered impractical with regard to real-world large-scale databases. Therefore, most recent approaches are hardware-based, i.e. they require tamper-proof hardware:

- In [104], a secure coprocessor is used as a tamper-proof device for storing and propagating the relevant database record, based on an encrypted query received from the user. As in earlier approaches, all records are read from the database, and therefore the computational complexity is still in $O(n)$.

- In [8], the previous approach is optimized by storing all original database records in encrypted and permutated form via the secure coprocessor. Thus, no longer all records have to read from the database, because observing the retrieval of an encrypted records does not give any information about the original record[4]. After a constant number of queries, the secure coprocessor switches to a different set of encrypted and permutated records. Because the preprocessing of encrypted databases in done off-line, i.e. independent of actual queries, on-line computa

Finally, there are the following approaches closely related to PIR:

[4]If only one record would be read for every query, however, subsequent queries would be revealed to be different if different encrypted records were read. Therefore, for the k-th query, all $k - 1$ records previously accessed have to be read again, and a random record has to be read for duplicate queries.

- *Private Information Storage*: The problem of Private Information Storage, i.e. the unobservable storage of data in a remote database, is similar to the problem of PIR, and analogous solutions have been suggested e.g. for multi-server schemes and hardware-based schemes. We omit a detailed discussion of these approaches here, because they are not directly relevant for the field of Privacy-Preserving Information Filtering.

- *Symmetrically-Private Information Retrieval*: If the privacy of the database provider has to protected in addition to the privacy of the user, a Symmetrically-Private Information Retrieval (SPIR) scheme is required. The privacy of the database is protected if a user does not receive any information in addition to the result of the query. Starting with [53], there are a number of approaches addressing this problem explicitly. In hardware-based PIR schemes, it is usually addressed implicitly, because only the query result is returned to the user anyway.

3.2.4 Privacy-Preserving IF Architectures

Related work in privacy-preserving IF architectures focuses on distributed architectures, i.e. on collaboration-based approaches. The main problem in distributed IF architectures is how to determine for a given user either the most similar users themselves or at least potential candidates for recommendations. In [85], the following five approaches are suggested:

- *Random Discovery*: A file-sharing protocol or similar mechanism is used for finding and contacting other users. This approach is not very efficient because a discovered user will only be similar coincidentally and therefore a large number of other users has to be contacted.

- *Transitive Traversal*: The Random Discovery approach is improved by the the following approach: A given user asks another user that is known to be similar for a list of further similar users, and contacts these, because they are expected to be similar to the given user as well. In this approach, a new user has still to contact other users randomly.

- *Centralized Model Generation*: A trusted central server stores all objects themselves, and all user ratings of these objects, in a single model. This approach is obviously critical with regard to privacy, because sensitive information is stored centrally and trust is required to a large extent unless the sensitive information is protected via other mechanisms.

- *Distributed Model Generation*: Using an approach similar to a secure voting scheme, such as a secure blackboard, a global model may be created in a privacy-preserving manner. Apart from the complexity, the main drawback of this approach is the fact that it cannot be used to determine similar users in addition to the recommendations themselves, because global models abstract from user-user relationships.

- *Distributed Model Storage*: In peer-to-peer networks with a deterministic overlay routing system, the network itself may be used as a distributed storage system for user-object relationships. This approach is problematic with regard to privacy as well, because each user has to make a list of rated objects available for other users.

A number of solutions have been suggested along these main approaches:

- Competitive Recommender Systems: In [9] a distributed algorithm based on the competitive recommender systems approach [41] is introduced that is based essentially on random discovery: A recommendation is generated for a user by iteratively retrieving an item and determining its rating as a recommendation, until an item with a sufficiently high rating is found. An item is retrieved either randomly from the provider, or by asking a random user for a recommendation, the decision being based on a random coin flip. The complexity of this algorithm is largely dependent on the assumption that the number of types is small or even constant with regard to the number of users, and on the assumption that the number of users belonging to a type is significantly larger than the number of users not belonging to any type, because otherwise a large average number of iterations is required for each user.

- Yenta: In [45], an agent-based approach is described in which user agents representing similar users are discovered via the transitive traversal approach. A hill-climbing algorithm is applied in order to find a user agent with maximal similarity. The problems inherent to this approach, namely the issues of local maxima and isolated groups of agents are mentioned, but, somewhat surprisingly, stated to be negligible. Privacy is preserved through pseudonymous interaction between the agents and adding obfuscating data to personal information, resulting in a "probable innocence" degree of unlinkability. More recent related approaches are described e.g. in [79].

- PocketLens: As described in [85], all five approaches are implemented as reference architectures for the PocketLens algorithm, a neighborhood-based algorithm for distributed collaborative filtering. The realized architectures are characterized by the respective drawbacks listed above.

- Alambic: The Alambic system [4] is based on a centralized model generation approach in which the privacy of the model is preserved by a combination of mechanisms. It is described in detail below.

- Cryptography-Based Collaborative Filtering: Various cryptography-based approaches [24][25] generating a global model in a distributed way have been suggested. They are described in detail below.

- Reconstruction-Based Collaborative Filtering: Various reconstruction-based approaches [93][92] generating a central model in a privacy-preserving way have been suggested. They are described in detail below.

Finally, it should be noted that other aspects not related to privacy have to be addressed as well when designing a distributed IF architecture: In [113], an agent-based distributed recommender system with the focus on utility rather than privacy is described, i.e. the main problem addressed is how to enable agents decide when and to whom to provide recommendations. It is concluded that agent with similar profiles may profit from exchanging recommendations, but no mechanism for determining likely candidates is given apart from random interaction of agents, which is rather inefficient. Privacy aspects are explicitly not addressed.

The approaches most relevant as work related to our approach are discussed in detail in the following.

3.2.4.1 Alambic

The Alambic system [4] proposes a mechanism for privacy-preserving demographic filtering which may be generalized in order to support other kinds of filtering techniques. While its description strongly suggests that the architecture is based on MAS technology, this is not stated explicitly. Basically, the filter is realized as an independent entity which is to some degree controlled by the provider (all communication with other entities is controlled and monitored by the provider, by unspecified means), but at the same time protected against manipulations (through code encryption and obfuscation, or alternatively by utilizing tamper-proof hardware). The provider model contains clusters of users with a similar demographic background, where each cluster is assigned indexes of the provider profile elements themselves. The provider model is exclusively controlled and accessed by the filter role. It is updated based on feedback of users.

In a filtering process, the user profile is encrypted via the filter role's public key as protection against the provider entity. The encrypted profile is sent from the user role to the provider entity, where it is stored for unspecified reasons. It is then propagated to the filter entity, where it is decrypted and subsequently compared with the cluster centroids in order to determine the best matching cluster. The indexes of profile elements associated with the best matching cluster are encrypted twice via a symmetric encryption scheme, first with a secret key shared with the provider entity (as protection against the user entity), and then with a secret key shared with the user entity via the encrypted profile (as protection against the provider entity).

The encrypted indexes are returned to the user entity, presumably again via the provider entity. Finally, the actual recommendations are determined via the indexes, either directly by the provider role itself, or via a more complex PIR-based protocol between user and provider entity which keeps the actual recommendations hidden from the provider. Again, details of the protocol are unspecified. All user interaction with the system is anonymous. The specific algorithm used for determining clusters is not specified, because it is irrelevant for the overall privacy-preserving architecture.

3.2.4.2 Cryptography-Based Collaborative Filtering

In [24], a distributed privacy-preserving architecture for a recommender system based on collaborative filtering via Singular Value Decomposition is described. In this approach, recommendations are generated via a public model aggregating the distributed user profiles without containing explicit information about user profiles themselves. A similar approach based on factor analysis is described in [25]. Both approaches are based on secure multi-party computation.

The overall complexity is $O(m\, n\, log\, n)$ for n users and m different items contained within the user profiles, which is reasonable close to the lower bound of $\Omega(m\, n)$. The model is structured as a matrix P with P_{ij} being the rating of user i for item j, which is zero if the item has not been rated, and greater than zero otherwise. This matrix is not known to any participant. Recommendations are generated via the public matrix A, computed via SMPC, with A being a global linear approximation of P.

All computation is distributed between the users. An honest majority of users is assumed, i.e. a fraction $\alpha > 1/2$ of all involved computers are assumed to be uncorrupted. The approach assumes two services: A blackboard in the form of a write-once, read-many storage system, and a trusted source of random bits. The protocol itself, which we do not present here, is largely based on a scheme for secure e-voting described in [35]. It uses homomorphic encryption[5] in order to handle sums of encrypted vectors without exposing the underlying data itself.

3.2.4.3 Reconstruction-Based Collaborative Filtering

In [93], a distributed privacy-preserving architecture for a recommender system based on collaborative filtering via Singular Value Decomposition is described, i.e. an architecture very similar to the one described above. The main difference is the following: Privacy is preserved through random perturbation instead of secure-multi party computation. A similar approach based on correlation of users is given in [92]. In the first approach, a global model is created based on vectors representing the single user profiles in which all data has been perturbed by adding random values to the vector elements, based on a given distribution. A significant trade-off between privacy and accuracy is observed: Obviously, using a small amount of perturbation is insufficient for preserving privacy because the modified vector closely resembles the original vector. On the other hand, a large amount of perturbation decreases the overall accuracy because the vector elements become indistinguishable from random noise. Determining the optimal trade-off is problematic because while accuracy can be expressed in terms of mean absolute error of the model created from the perturbed data, privacy is generally more difficult to quantify. Based on the method suggested in [1], a privacy measure is used indicating how closely an original value can be estimated based on the perturbed value: The *privacy loss* $P(X|Z)$ is the fraction of privacy of a variable X that is lost by revealing the variable Z. Privacy of a variable itself is expressed based on its differential entropy, i.e. a measure of uncertainty in the values of the variable. Using these measures for privacy and accuracy, an optimal trade-off may be determined.

3.2.5 Evaluation

Unlike single PETs, the Privacy-Preserving Technologies discussed in this section are sufficient for preserving privacy in specific application domains, sometimes under certain realistic assumptions. While there is a comparatively small amount of related work describing PPTs for Information Filter-

[5]Homomorphic encryption schemes basically allow operations on encrypted data to be carried out in a meaningful manner.

ing, the related approaches discussed here are also relevant in the context of PPIF because they may used as parts of the overall architecture:

- Peer-oriented approaches methods may be used as part of a filtering technique in collaboration-based approaches, i.e. for clustering users in order to determine similar users. While feature-based approaches may also be based on clustering of profile elements, in this case the clustering itself generally does not have to be performed in a privacy-preserving manner because the underlying information does not refer to more than one participant.

- Privacy-preserving data mining approaches are relevant because data mining methods may be used for creating provider profile models in collaboration-based approaches, e.g. by deriving association rules based on user feedback.

- Private Information Retrieval schemes are intrinsically very relevant in the context of PPIF because the filtering process may rely on queries on a large dataset. Apart from performance issues, however, proposed solutions in this area are currently of a more theoretical nature and cannot easily be applied to real-world database management systems.

However, these approaches may not be used by themselves for realizing Privacy-Preserving Information Filtering, because the respective IF system would not meet all requirements listed in Section 2.3.4:

- By definition, Peer-oriented approaches methods are only suitable for collaboration-based approaches and thus do not even meet all functional requirements. Even ignoring this aspect, they do not address filter privacy, and are characterized by a high complexity, leading to problems with regard to performance and/or broadness.

- Privacy-preserving data mining approaches do not address filter privacy, and protect the privacy of the user only up to the information processing stage, but not in the information filtering stage. They are additionally characterized by quality issues (perturbation-based approaches) or performance issues (approaches based on Secure Multi-Party Computation).

- Private Information Retrieval schemes do not address provider privacy in the context of IF, nor filter privacy. As largely theoretical approaches, they do not meet the requirement of adequate performance in realistic scenarios.

Acceptance is achieved to a large degree by meeting the respective privacy requirement.

Table 3.2 summarizes the evaluation of these PPTs in the context of all work related to PPIF. To recapitulate, the PPTs discussed in this section provide useful functionality for parts of PPIF architectures but cannot be used by themselves in order to realize these architectures.

Finally, we evaluate the existing approaches for distributed PPIF architectures. These approaches are usually restricted to specific filtering techniques and are only suitable for collaboration-based approaches and thus do not meet all functional requirements. With regard to the non-functional requirements, they are characterized by the following drawbacks:

- Non-model-based approaches, such as the competitive recommender systems approach and Yenta, are infeasible for real-world applications, mainly due to the complexity involved in the task of determining similar users. They do not address filter privacy, and are restricted to specific filtering techniques, which is problematic with regard to the requirement of broadness.

- The *PocketLens* approach is characterized by the drawbacks listed in the discussion of the five main approaches for determining similar users, which are problematic either with regard to user privacy, or with regard to performance. Furthermore, the *PocketLens* approach does not address filter privacy.

- The *Alambic* system is the approach most closely related to our architecture. While basically viable, two aspects are insufficiently addressed, namely the protection of the filter against manipulation attempts (the technology-based solutions are only described in an abstract way, while a solution based on an additional trusted party only moves the underlying problem to a different level), and the prevention of collusions between the filter and the provider (the suggested solution again relies on a trusted third party). In other words, the approach does not address filter privacy, and it only insufficiently addresses user privacy.

- All *Cryptography-Based Collaborative Filtering* approaches are largely theoretical, i.e. they often have not even been fully implemented. Furthermore, they require the participation of all users, especially for updating the model, a course of action that is somewhat inefficient for a large number of users. Additionally, the suggested approaches cannot be used to realize a Matchmaker System, because similar users are not determined as such, and they do not address filter privacy.

- The described *Reconstruction-Based Collaborative Filtering* approach deals only with the privacy-preserving generation of the model, it does not address the problem of obtaining recommendations in a privacy-preserving manner. Therefore, only the information procession stage of an entire IF process is covered. As in the case of perturbation-based approaches for privacy-preserving data mining, the requirement of quality is problematic, and, as in all approaches discussed in this section, filter privacy is not addressed.

Acceptance is achieved to a large degree by meeting the respective privacy requirement. Provider acceptance of these approaches is expected to be somewhat lower, though, because these approaches may not allow the provider to obtain any information about users at all.

Table 3.2 summarizes the evaluation of these distributed PPIF architectures in the context of all work related to PPIF. To recapitulate, none of the distributed PPIF architectures discussed in this section, which meet the non-functional requirements to a larger extend than single PETs or PPTs, constitutes a generic framework suitable for various kinds of filtering techniques and for realizing feature-based and collaboration-based Recommender Systems as well as Matchmaker Systems.

3.3 Privacy in Multi-Agent Systems

In this section, we discuss work related to privacy aspects of architectures based on Multi-Agent System technology. After a brief overview of approaches for anonymous communication between agents, we describe proposed solutions dealing with the problem of malicious hosts in MAS architectures.

3.3.1 Anonymous Communication

A MAS architecture requiring anonymous communication of agents may adapt general solutions for anonymous communication, as described in Section 3.1.1. Theoretic foundations for reasoning about anonymity and information hiding in MAS architectures are given by [61]. Furthermore, the following MAS-specific approaches for anonymous communication have been suggested:

- In [73], an approach for anonymous communication of mobile agents based on onion routing (see Section 3.1.1.1) is described which is is based on the JADE[6] multi-agent platform. Every agent platform consists of dedicated *onion agents* providing a data forwarding service (thus representing the mixes) which communicate via an Agent Communication Language, and an additional manager agent monitoring these agents.

- In [15], the anonymizing service provided by the AgentScape[7] framework is described. It is realized as a simple router, i.e. a proxy agent (see Section 3.1.1.1) relaying communication between two agents that intend to remain anonymous.

Both approaches may be adapted to other MAS architectures in a straightforward manner.

3.3.2 Protection against Malicious Hosts

As discussed in Section 2.4.3, the protection of agents against malicious hosts is a central requirement in MAS architectures consisting of personal agents representing different entities that regard each other as potentially non-honest. The solutions addressing this problem have been suggested:

- *Code encryption*: As described in [98], the code of a mobile agent my be protected by using an encrypted function computing the result for input provided by the host. While the host obtains the result of the function, he is not able to obtain the original function itself. He may, however, still attack the mobile agent by re-running it and attempting to obtain information about the function by using different inputs. These kinds of attacks may be prevented by requiring communication with a trusted third party during the computation of the result [5].

- *Data Encryption*: In addition to encrypting a function as part of a mobile agent's algorithm, the data of an agent may be encrypted as well, and all operations of the agent on an untrusted platform may be carried out on the encrypted data itself, without the need for decrypting the data. This approach requires a homomorphic encryption scheme, e.g. a scheme based on the ElGamal cryptosystem [42].

- *Code Obfuscation*: Code obfuscation in the context of MAS is the transformation of an agent's code into a form that is hard to understand or

[6]Available via the URL <http://jade.tilab.com/>.
[7]Available via the URL <http://www.agentscape.org>.

to reverse-engineer by attackers, but at the same time results in equivalent behavior of the agent executing the obfuscated code, compared to an agent executing the original code. A comprehensive overview is given e.g. in [37]. It has been shown [10] that there is no general obfuscation method that always results in a perfectly obfuscated program which reveals nothing about the underlying algorithm. Research in the area continues, however, because in practice less-than-ideal obfuscation may be acceptable: In the case of mobile agents, an obfuscation scheme deterring or delaying attacks for a certain amount of time could still be useful for protecting mobile agents against manipulations while they are running on an untrusted platform.

- *Recording and Tracing Approaches*: A number of other approaches have been suggested which mainly aim at detecting malicious hosts, without the possibility of preventing attacks. A survey of these approaches is given in [64]. They are based on recording the itinerary of mobile agents visiting a number of mobile platforms, and detecting possible malicious hosts by inconsistencies in the recordings [97], or by tracing the execution of an agent in a non-repudiable log file [114].

- *Trusted Computing*: While the approaches outlined above may be sufficient in specific scenarios, the only approach suitable for protecting mobile agents against malicious hosts in a generic way appears to be the use of secure hardware, e.g. through trusted computing, as described in Section 3.1.3. Via remote attestation, a platform may be ensured to be incapable of acting in a malicious way, e.g. by realizing the virtual machine approach for semantic remote attestation described in [60].

The drawbacks of these approaches are discussed in the following section.

3.3.3 Evaluation

While the solutions for anonymous communication in MAS architectures constitute adequate solutions, the proposed approaches dealing with the problem of malicious hosts are characterized by several drawbacks:

- *Code encryption*: The main drawback of this approach is that mainly because of complexity issues, it can be applied efficiently only to basic algorithms, such as algorithms for evaluating polynomial expressions. As in the case of secure multi-party computation (see Section 3.1.2, it is infeasible for complex algorithms such as filtering techniques for Information Filtering.

- *Data Encryption*: Data encryption in itself is sufficient for protecting data of a mobile agent, as long as the data is static in the following sense: The data should be obtained before the agent migrates to the untrusted platform, and decrypted only after the agent has left it. Unencrypted data received through communication while the agent is running on the untrusted platform should not be combined with the encrypted data in any way. This restriction implies that most mobile agent scenarios cannot use this approach, because they usually involve communication of the mobile agent, except for simple remote computing scenarios.

- *Code Obfuscation*: Code obfuscation in itself is insufficient for protecting the privacy of mobile agents against non-honest hosts, because the respective attacks are not limited to the duration of the execution of the mobile agent.

- *Recording and Tracing Approaches*: These approaches do not actually protect the privacy of mobile agents, which makes them infeasible for many scenarios including Privacy-Preserving Information Filtering.

- *Trusted Computing*: As described above, trusted computing appears to be the only approach suitable for protecting mobile agents against malicious hosts in a generic way.

To recapitulate, privacy aspects in MAS architectures are largely covered by related work. Therefore, we do not provide additional approaches addressing these aspects in this work, but rather build upon the existing solutions.

3.4 Summary

This chapter discusses different areas of related work, namely Privacy-Enhancing Technologies (Section 3.1) and Privacy-Preserving Technologies (Section 3.2), including approaches for distributed Privacy-Preserving Information Filtering. It also discusses work related to aspects of privacy in Multi-Agent Systems (Section 3.3). We summarize the evaluation of the various areas of work related to PPIF (i.e. Section 3.1.5 and Section 3.2.5) in Table 3.2, based on the non-functional requirements defined in Section 2.2.3. To recapitulate, no single solution is sufficient for realizing an architecture for Privacy-Preserving Information Filtering meeting all functional and non-functional requirements.

Table 3.2: An overview of existing PETs, PPTs, and approaches for PPIF in relation to the requirements and acceptance aspects of Privacy-Preserving Information Filtering. A requirement may be fully met (indicated by "✓"), partially met (indicated by "o"), or not met at all (indicated by "–"). Acceptance is indicated in an analogous manner. Note that the ratings do not always indicate the best value theoretically possible for the architecture utilizing the respective technology, or the respective architecture for distributed PPIF, but a realistic average value.

	Privacy Requirements			Other Requirements			Acceptance	
	R_u	R_p	R_f	R_{qq}	R_{bb}	R_{pp}	A_u	A_p
PETs								
Anonymous Communication	–	✓	–	o	o	o	✓	–
SMPC	✓	✓	–	o	–	–	o	o
Trusted Computing	✓	✓	✓	o	–	o	–	o
Privacy Enforcement	–	✓	–	o	o	o	o	o
PPTs								
Peer-Oriented Approaches	o	✓	–	o	–	–	o	o
SMPC-based PPDM	–	✓	–	o	–	–	–	✓
Perturbation-based PPDM	–	✓	–	–	–	o	–	✓
Private IR	✓	o	–	o	o	–	o	o
Distributed PPIF								
Random Discovery-based	o	✓	–	o	–	–	o	o
PocketLens	o	✓	–	o	–	–	o	o
Alambic	o	✓	–	o	–	o	o	o
Cryptography-based	✓	✓	–	o	–	–	o	o
Reconstruction-based	✓	✓	–	–	–	o	o	o

At first glance, out of all related work, trusted computing seems to be the most suitable approach for PPIF. Consequently, a straightforward course of action would be to take an existing comprehensive IF system and to the respective architecture based on trusted computing. As discussed in Section 10.2, this approach is problematic mainly with regard to the requirement of broadness and the aspect of user acceptance. Therefore, even if the IF system is to be based on trusted computing, a different approach is required. We describe this approach in the following chapter.

Chapter 4

Privacy-Preserving Information Filtering

This chapter gives an overview of our approach for Privacy-Preserving Information Filtering. It focuses on the general idea and main concepts of our solution, and omits all details of the approach, which are covered by the following chapters.

This chapter is structured as follows: The following section lists the supported use cases, which are directly derived from the functional requirements listed in Section 2.3.4. Section 4.2 gives a high-level outline of our solution for PPIF. Based on two essential concepts, namely the concept of a trusted environment for protecting privacy in Recommender Systems (Section 4.2.1), and the additional concept of an anonymous centralized model for protecting privacy in Matchmaker Systems (Section 4.2.2), we motivate the use of MAS technology in this context (Section 4.2.3) and introduce the components of our approach that realize these concepts via MAS technology (Section 4.2.4). In Section 4.3, we briefly describe the implementation of the approach itself as well as the implementation of a prototypical application based on the approach. Section 4.4 summarizes the chapter.

4.1 Use Cases

As defined in Section 2.2.1, we distinguish between the following kinds of IF systems, each fulfilling a distinct goal:

- A *Recommender System* generates recommendations and predictions of items.
- A *Matchmaker System* determines similar users.

- A *Hybrid IF System* generates recommendations and predictions of items via determining similar users.

A comprehensive IF approach should provide functionality that is sufficient for realizing these kinds of systems. In other words, an approach for PPIF should provide functionality for realizing the following four main use cases:

- The use case “get prediction for item” with interactions resulting in $pred_{u,s,ft,i}$, i.e. the predicted relevance of item i for a given user u, based on the profile of the supplier s (which may be a provider or another user), and the filtering technique ft.

- The use case “get recommendations” with interactions resulting in $REC_{u,s,ft,n}$, i.e. the top-n recommendations with parameters as above.

- The use case “get prediction for user” with interactions resulting in $pred_{u,s,ft,u'}$, i.e. the predicted similarity of user u' with parameters as above.

- The use case “get similar users” with interactions resulting in $SU_{u,s,ft,n}$, i.e. the top-n similar users with parameters as above.

We note that each use case actually consists of two partial use cases, one in which the complete supplier profile is used, and one in which a constrained supplier profile is used, containing elements returned as a result of a query on the supplier profile: The latter use case constitutes the mixed IR/IF scenario, the IR-related part of which is not considered as privacy-critical, because it is assumed not to be directly related to the user profile data. A plausible scenario for this use case is the following example: A user wants to receive recommendations from a movie recommender, but the recommendations should be restricted to movies being shown on a specific day on television or in cinemas in a specific city. In this case, the respective queries usually return a more manageable part of the supplier profile, such as 200 out of 20.000 movies.

Additionally, these use cases may be split up further depending on the actual degree of user privacy (see Section 2.3.3.3), which is basically defined via the result data the supplier obtains:

- *Completely Linkable Result Data*: All result data is linkable to the user. Consequently, it is internally linkable as well.

- *Semi-Linkable Result Data*: The result data is internally linkable, but not to the user. As an example, the supplier may determine whether two recommendations belong to one set of recommendations generated for a single user, but the user himself remains anonymous. This case does not apply to the prediction-based use cases, because the concept of unlinkability does not apply to single elements.

- *Semi-Private Result Data*: The result data is internally unlinkable, and it is not linkable to a user as well.
- *Completely Private Result Data*: The supplier does not obtain any result data.

We subsume the first two cases under the designation *Linkable Result Data*, and the last two cases under the designation *Private Result Data*.

In addition to these use cases, a comprehensive approach for PPIF should also provide functionality for the following use cases related to the first two stages of an IF process, as described in Section 2.2.1:

- The use case "update profile elements", in which an entity adds elements to the respective profile, or removes elements from it during the Information Collection stage.
- The use case "update profile model", in which the model for a profile is updated based on added or removed elements during the Information Processing stage.

Table 4.1 summarizes all six use cases described above, which we refer to as *main use cases* in the following. Other use cases providing aggregated by these main use cases are referred to as *partial use cases*. As they are not directly relevant for the outline of our solution, they are not listed here.

4.2 Outline of the Solution

We propose an approach for agent-based Privacy-Preserving Information Filtering suitable for realizing Recommender Systems, Matchmaker Systems, and Hybrid IF Systems. The approach meets all requirements stated in Section 2.3.4 and may be used to realize all use cases introduced above, as shown in Section 10.1.

In the following outline of our solution, we focus on the information filtering stage and disregard the two preceding stages, mainly because they are no more critical with regard to privacy as they typically involve fewer entities:

Table 4.1: Main use cases covered by our approach for PPIF.

main use case	supplier profile	result data		
		compl. linkable	semi-linkable	private
information collection stage				
update profile elements	n/a	n/a (no result data)		
information processing stage				
update profile model	n/a	n/a (no result data)		
information filtering stage				
get prediction for item	complete	✓	n/a	✓
get prediction for item	query result	✓	n/a	✓
get recommendations	complete	✓	✓	✓
get recommendations	query result	✓	✓	✓
get prediction for user	complete	✓	n/a	✓
get prediction for user	query result	✓	n/a	✓
get similar users	complete	✓	✓	✓
get similar users	query result	✓	✓	✓

- A process of the information collection stage only involves a single entity, namely the user or the provider entity, depending on the profile data to be collected.

- A feature-based process of the information processing stage involves two entities, namely the user or the provider entity, depending on the profile data to be processed, and the filter entity. As noted in Section 2.2.1, these processes occur in Recommender Systems only.

- A collaboration-based process of the information processing stage typically involves all three abstract entities. As noted in Section 2.2.1, these processes occur in Matchmaker Systems and Hybrid IF Systems.

- A process of the information filtering stage involves all three abstract entities.

Additionally, processes of the first two stages do not return sensitive result data, as in the final stage. Therefore, the concepts of a solution for the information filtering stage are likely to be applicable for the other stages as well (Section 7.2.2.1 and Section 7.2.2.2 show that this is in fact the case).

We begin the outline by addressing the most problematic aspect in Privacy-Preserving Information Filtering, namely the apparent paradox of providing private information in order to obtain personalized information without losing control over the provided private information.

4.2.1 Trusted Environment

The requirements state that no private information should be acquired permanently by other entities. The basic idea for realizing the privacy-related requirements in Recommender Systems is already suggested implicitly by the use of the adverb "permanently" in this context: While it is obviously important that permanent acquisition is prevented, temporary acquisition of private information may be allowed and therefore used to full capacity. Thus, the use case "get recommendations" is realized as follows on the most abstract level: User and supplier entity both propagate the respective profile data to the filter entity. The filter entity provides recommendations (either to both entities, or only to the user entity), and deletes all private information afterwards.

There are basically two approaches for realizing an acquisition of private information that is in fact only temporary:

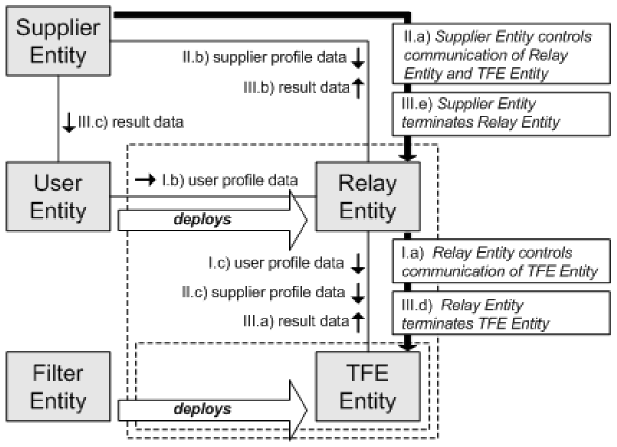
- *Trusted Software*: The respective entity is trusted or known - e.g. through validation via trusted computing mechanisms - to remove the respective information as specified;
- *Trusted Environment*: The respective entity operates in an environment that is trusted or otherwise known to control the communication and lifecycle of the entity to an extent that the removal of the respective information may be achieved regardless of the attempted actions of the entity itself. Additionally, the environment itself is trusted or otherwise known not to act in a malicious manner in this context (e.g. it cannot extract and propagate the respective information itself).

Our solution is based on a trusted environment because, although it is more complex than the trusted software approach, the trust issues are resolvable more easily in this approach, basically because a trusted environment may be realized in a more generic way. We address this issue in detail in Chapter10.2.

According to this decision, we specify the abstract information filtering protocol for the use case "get recommendations (linkable result data)" as shown in Figure 4.1: The filter entity deploys a *Temporary Filter Entity (TFE)* operating in a trusted environment. The user entity deploys an additional relay entity operating in the same environment. These additional entities are short-lived, i.e. they are terminated after a specified number of tasks. Through mechanisms provided by this environment, the relay entity is able to control the communication of the TFE, and the supplier entity is able to control the communication of both relay entity and the TFE. Thus, it is possible to ensure that the controlled entities are only able to propagate recommendations, but no other private information.

In the first stage (Step *I.a* to Step *I.c* of Figure 4.1), the relay entity establishes control of the TFE, and thus prevents it from propagating user profile information. User profile data is propagated without participation of the supplier entity from the user entity to the TFE via the relay entity. In the second stage (Step *II.a* to Step *II.c* of Figure 4.1), the supplier entity establishes control of both relay and TFE, and thus prevents them from propagating supplier profile information. Supplier profile data is propagated from the supplier entity to the TFE via the relay entity. In the third stage (Step *III.a* to Step *III.e* of Figure 4.1), the TFE returns the recommendations via the relay entity, and the controlled entities are terminated. Taken together, these steps ensure that all private information is acquired temporarily only by the other main entities. The use case "get prediction for item (linkable result data)" is realized via a similar protocol, in which the result data contains the prediction instead of recommendations.

The use cases "get recommendations (private result data)" and "get prediction for item (private result data)" are realized by a protocol based on similar steps (see Figure 4.2), but including an additional relay entity deployed by the supplier, which is basically required for validating the result data before it is propagated to the user. In the case of linkable result data, this task is carried out by the supplier entity itself, which is not possible in the case of private result data, because the supplier entity may not obtain result data directly.



Supplier Entity
II.b) supplier profile data ↓
III.b) result data ↑
II.a) Supplier Entity controls communication of Relay Entity and TFE Entity
III.e) Supplier Entity terminates Relay Entity
↓ III.c) result data
User Entity
→ I.b) user profile data
deploys
Relay Entity
I.a) Relay Entity controls communication of TFE Entity
I.c) user profile data ↓
II.c) supplier profile data ↓
III.a) result data ↑
III.d) Relay Entity terminates TFE Entity
Filter Entity
deploys
TFE Entity

Figure 4.1: The abstract privacy-preserving information filtering protocol for the use cases returning linkable result data. All communication across the environments indicated by dashed lines is prevented with the exception of communication with the controlling entity.

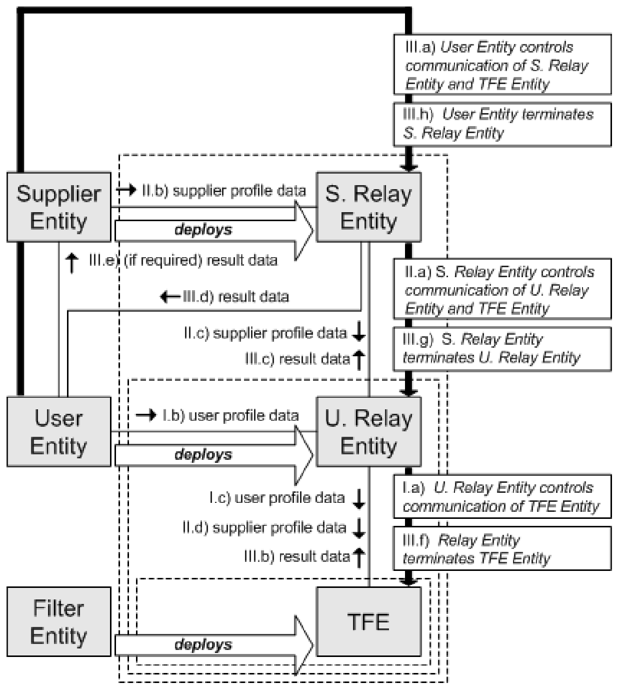

III.a) User Entity controls communication of S. Relay Entity and TFE Entity
III.h) User Entity terminates S. Relay Entity
Supplier Entity
II.b) supplier profile data
deploys
S. Relay Entity
III.e) (if required) result data
III.d) result data
II.a) S. Relay Entity controls communication of U. Relay Entity and TFE Entity
II.c) supplier profile data
III.c) result data
III.g) S. Relay Entity terminates U. Relay Entity
User Entity
I.b) user profile data
deploys
U. Relay Entity
I.a) U. Relay Entity controls communication of TFE Entity
I.c) user profile data
II.d) supplier profile data
III.b) result data
III.f) Relay Entity terminates TFE Entity
Filter Entity
deploys
TFE

Figure 4.2: The abstract privacy-preserving information filtering protocol for the use cases returning private result data. All communication across the environments indicated by dashed lines is prevented with the exception of communication with the controlling entity.

While these protocols may also be applied in distributed Matchmaker Systems, in this case another central problem has yet to be addressed, namely the challenge of determining user candidates in an efficient manner.

4.2.2 Anonymous Centralized Model

In order to meet the privacy-related requirements in Matchmaker Systems, the protocols introduced above may be applied, with a different user constituting the supplier entity in the interactions. Determining similar users in general, however, is difficult if the number of users is too large to efficiently carry out this protocol for each pair of users. As described in Section 3.2.4, there are various approaches for determining suitable candidates from the set of all users. Our solution is based on a combination of the mechanisms of random discovery, transitive traversal, and central model generation.

In order to preserve privacy, all information related to users is stored anonymously in a centralized model: A user adding a specific item to his profile may announce this anonymously, which allows the provider of the centralized model to store the relationship of item and pseudonym. By using a different pseudonym for each user-element relation stored in the central model, unlinkability of users and items as well as of the items among themselves is realized. Subsequently, a given user may obtain the information that the profiles of other users contain a specific item, but is given only a pseudonymous communication address for contacting the candidate user. Obviously, a mechanism for anonymous communication is required for this solution.

The provider of the centralized model does not necessarily have to be identical with the provider of the underlying data. In most scenarios, however, a single entity is likely to constitute both providers, because maintaining a centralized model allows the provider to obtain some feedback regarding the prevalence of items in user profiles, which may be useful information. Other entities are less likely to be sufficiently motivated to provide a centralized model.

The protocol for the use case "get similar users" is defined as follows: The user entity anonymously receives anonymous candidates from the provider, which may be selected randomly or based on his profile elements. The user entity interacts with the candidates in order to determine the similarity of the respective profiles, or in order to obtain additional candidates with whom he interacts in the same manner. Over time, the most similar users are found with high probability. The user may also receive candidate users randomly, or from other similar users.

4.2.3 Use of MAS Technology

As outlined in the previous sections, our approach for Privacy-Preserving Information Filtering is based on a distributed system in which the main abstract entities of user, provider and filter are modeled as distinct entities which control the respective private information exclusively. Thus, only interactions involving more than one entity have to be considered as privacy-critical. If communication control as described above is actually realized, and the entities are protected against external threats, sensitive information may actually be protected and a PPIF architecture may be realized. This approach requires a participating entity to have the following five main abilities:

- The ability to perform certain well-defined tasks (such as carrying out a filtering process) with a high degree of autonomy, i.e. largely independent of other entities (e.g. because the entity is not able to communicate in an unrestricted manner);

- The ability to be deployable dynamically in a well-defined environment;
- The ability to communicate with other entities;
- The ability to achieve protection against manipulation attempts;
- The ability to control and restrict the communication of other entities;

As defined in Section 2.4.1, MAS architectures are an ideal solution for realizing the approach, because they provide agents as entities actually characterized by autonomy, mobility and the ability to communicate, as well as agent platforms as environments providing means to realize the security of agents. In this context, the issue of malicious hosts, i.e. host attacking agents, has to be addressed explicitly. Additionally, existing MAS architectures generally do not allow agents to control the communication of other agents, i.e. this specific ability is not covered as such. It is possible, however, to expand MAS architecture in order to provide agents with this ability. For these reasons, our approach is based on a MAS architecture. Concluding the outline, we give a high-level overview of the architecture and lists its main components.

4.2.4 Main Components

Continuing the depictions of existing IF architectures in Section 2.2.2, Figure 4.3 shows a high-level overview of the architecture for PPIF for the non-collaboration-based scenario. In addition to the MAS architecture itself, which is assumed as given, the architecture in general consists of the following five main components providing the required functionality:

- Because MAS architectures generally do not provide functionality for controlling the communication of agents, nor for anonymous communication, a component realizing this functionality is provided, namely the *Infrastructure Module* described in Chapter 5.
- In order to facilitate the use of different data storage mechanisms, and to provide a uniform interface for accessing persistent information, which may be utilized for monitoring critical interactions involving potentially private information e.g. as part of queries, a component for transparent persistence is provided, namely the *TPMAS Module* described in Chapter 6.

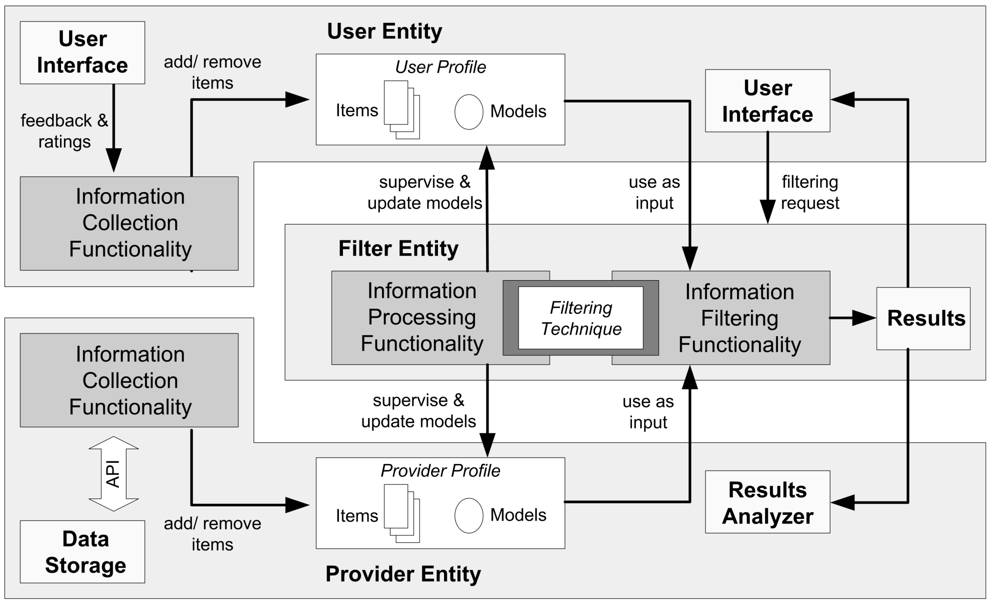


Figure 4.3: The proposed architecture for Privacy-Preserving Information Filtering.

- Functionality for realizing the Recommender System use cases "get prediction for item" and "get recommendations" is provided within the *Recommender Module* component described in Chapter 7.
- Functionality for realizing the Matchmaker System use cases "get prediction for user" and "get similar users" is provided within the *Matchmaker Module* component described in Chapter 8.
- Finally, while these components may generally be used in connection with various filtering techniques that are not restricted to specific domains, the protocols impose certain other restrictions on the actual filtering techniques. Therefore, *Exemplary Filtering Techniques* are provided as a separate component described in Chapter 9 in order to show that the requirements may actually be met by choosing appropriate filtering techniques.

Figure 4.4 gives an overview of the five main components.

All non-functional requirements listed in Section 2.3.4 are addressed by these components, with different components focusing on different requirements:

- The requirement of user privacy is primarily addressed by the Infrastructure Module providing foundations for privacy protection, the

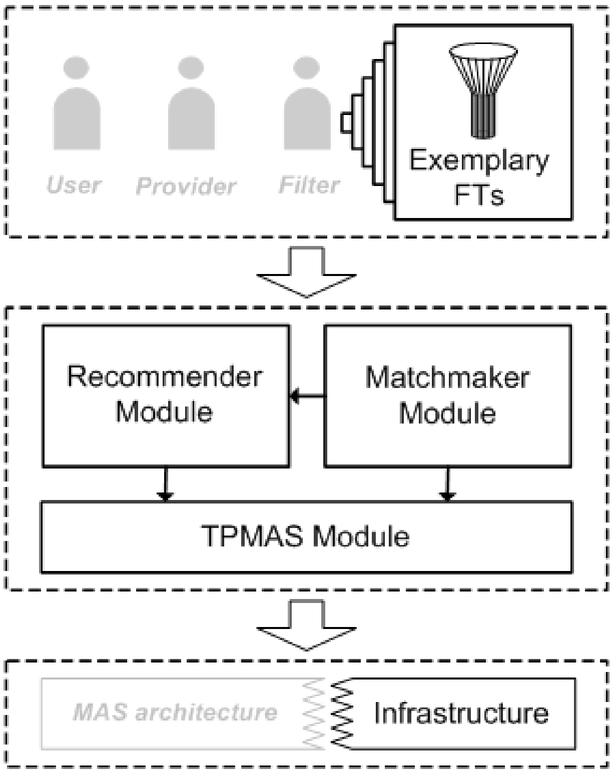


Figure 4.4: The five main components of the PPIF architecture. Additional parts required for the architecture but not directly contributed by this work are grayed out.

Recommender Module providing privacy-preserving protocols, and the Matchmaker Module protecting the privacy of users as participants in a distributed Matchmaker System.

- The requirement of provider privacy is primarily addressed by the Infrastructure Module and the Recommender Module, analogous to user privacy.

- The requirement of filter privacy is primarily addressed by the Infrastructure Module and the Recommender Module, analogous to user privacy.

- The requirement of quality is addressed by the Exemplary Filtering Techniques providing result data, and the Matchmaker Module providing users that are probably similar to a given user. In both cases, the provided information should be of high quality.

- The requirement of broadness is addressed by the TPMAS Module providing a uniform interface for accessing persistent information, and by Exemplary Filtering Techniques which are not restricted to a specific domain.
- The requirement of performance is primarily addressed by the Exemplary Filtering Techniques, because the filtering process itself is most critical with regard to performance. All other components take this requirement into account as well.

The algorithms used in the Exemplary Filtering Techniques component take the privacy requirements into account as well. Table 4.2 gives an overview of these relationships.

The trusted environment introduced above encompasses the MAS architecture itself and the Infrastructure Module. In other words, these components have to be trusted to act in a non-malicious manner to rule out the possibility of malicious hosts. Explicit trust with regard to the other components is not required because they operate within the trusted environment and are thus prevented from acting in a malicious manner. Finally, it should be noted that while we have chosen a specific MAS architecture for the implementation, the specification of the approach is applicable to any FIPA-compliant MAS architecture.

Table 4.2: The five main components of our approach for PPIF in relation to the requirements. A components provides primary functionality (indicated by "✓"), auxiliary functionality (indicated by "o"), or no explicit functionality (indicated by "–") with regard to a specific requirement.

	Privacy Requirements			Other Requirements		
	R_u	R_p	R_f	R_{qq}	R_{bb}	R_{pp}
Infrastructure Module	✓	✓	✓	–	–	o
TPMAS Module	–	–	–	–	✓	o
Recommender Module	✓	✓	✓	–	–	o
Matchmaker Module	✓	–	–	✓	–	o
Exemplary FTs	o	o	o	✓	✓	✓

4.3 Implementation

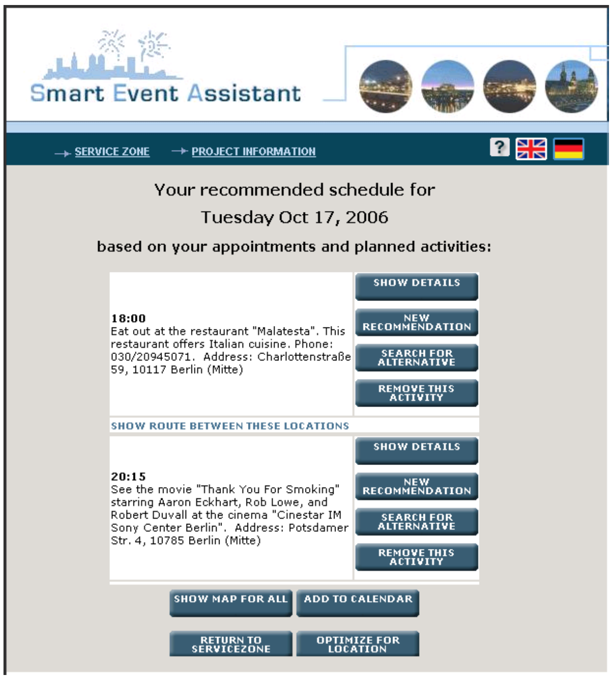
We have implemented our approach for Privacy-Preserving Information Filtering based on JIAC IV [50, 101, 100], a FIPA-compliant MAS architecture.

JIAC IV integrates fundamental aspects of autonomous agents regarding pro-activeness, intelligence, communication capabilities and mobility by providing a scalable component-based architecture. Additionally, JIAC IV offers components realizing management and security functionality, and provides a methodology for Agent-Oriented Software Engineering. In the context of PPIF, JIAC IV stands out among other MAS architectures as the only security-certified architecture, as it has been certified by the German Federal Office for Information Security according to the Evaluation Assurance Level 3 of the Common Criteria for Information Technology Security standard [52].

JIAC IV offers several security features in the areas of access control for agent services, secure communication between agents, and low-level security based on Java security policies [99]. Access control for agent services is based on authenticated users or X.509 certificates associated with agents. JIAC IV offers also means to secure the communication channel between agents. This is either achieved by using the SSL protocol on the transport level or, if this not possible, e.g. because a FIPA-compliant exchange of speech acts via the Agent Communication Channel is required, by using an application level protocol similar to SSL in order to protect speech acts. X.509 certificates are used for access control and for protecting the communication channel, based on a public key infrastructure [17]. Finally, Java security mechanisms [58] are used to protect agents from attacks performed by other agents within the same Java Virtual Machine. Java security mechanisms are also used to represent human users as subjects within the Java Authentication and Authorization architecture [76].

We have implemented all components listed above, following the decisions made in the analysis and design phase, which are described in the following chapters. As a proof of concept, and in order to evaluate performance and quality under real-life conditions, we have also used our approach within the Smart Event Assistant, a MAS-based Recommender System which integrates various personalized services for entertainment planning in different German cities, such as a restaurant finder and a movie finder [117]. Additional services, such as a calendar, a routing service and news services complement the information services. An intelligent day planner integrates all functionality by providing personalized recommendations for the various information services, based on the user's preferences and taking into account the location of the user as well as the potential venues. All services are accessible via mobile devices as well[1]. Figure 4.5 shows a screenshot of the intelligent

[1]The Smart Event Assistant was accessible online in different versions until 2007 via the URL <http://www.smarteventassistant.de>. It is currently being redeveloped and extended as the Smart Personal Assistant, which is accessible online via the URL <http://www.smartassistantsolutions.de>.



Your recommended schedule for
Tuesday Oct 17, 2006
based on your appointments and planned activities:

18:00
Eat out at the restaurant "Malatesta". This restaurant offers Italian cuisine. Phone: 030/20945071. Address: Charlottenstraße 59, 10117 Berlin (Mitte)

20:15
See the movie "Thank You For Smoking" starring Aaron Eckhart, Rob Lowe, and Robert Duvall at the cinema "Cinestar IM Sony Center Berlin". Address: Potsdamer Str. 4, 10785 Berlin (Mitte)

Figure 4.5: Screenshot of the Smart Event Assistant, a privacy-preserving Recommender System supporting users in planning entertainment-related activities.

day planner's result dialog. The Smart Event Assistant is entirely realized as a MAS system providing, among other functionality, various filter agents and different service provider agents, which together with the personal user agents utilize the functionality provided by our approach.

We describe typical scenarios of the Smart Event Assistant in more detail in Section 10.1.2.4, where we evaluate the performance of our approach. Due to resource restrictions with regard to the Smart Event Assistant project, we did not have the time to deploy the system in a trusted environment, i.e. based on a trusted computing infrastructure, which therefore remains future work.

4.4 Summary

This chapter gives an overview of our approach for Privacy-Preserving Information Filtering. From the requirements listed in Section 2.3.4, we derive a number of use cases which have to be realized by a comprehensive architecture for PPIF (Section 4.1). We outline our solution for PPIF, which is based on a trusted environment that is used to control the communication capabilities of entities deployed in this environment. Based on this trusted environment, we specify protocols for realizing the Recommender System-related use cases in a privacy-preserving manner (Section 4.2.1). We describe an anonymous centralized model of user-item relationships which is used for realizing the Matchmaker System-related use cases in a privacy-preserving manner (Section 4.2.2). We list the required abilities of entities operating in this context, and motivate the use of MAS technology by mapping these required abilities to the capabilities of agents and agent platforms (Section 4.2.3). We list the components of our approach, which address the given requirements (Section 4.2.4). We give a short overview of JIAC IV as the foundation of our implementation, and introduce the Smart Event Assistant as a prototypical application utilizing our approach (Section 4.3). The following chapters describe the components of our approach in detail.

Chapter 5

Basic Infrastructure

This chapter describes basic functionality for controlling the communication capabilities of agents, and functionality for anonymous communication of agents. We subsume both kinds of functionality in a single chapter because in both cases communication capabilities of agents are addressed, and because the functionality may be realized most efficiently by extending the respective MAS architecture itself.

The chapter is structured as follows: Section 5.1 briefly motivates the Infrastructure Module. Section 5.2 describes the ontologies, roles and interactions of the module, while Section 5.3 describes the agents and agent services realizing these interactions. Section 5.4 concludes the chapter with a summary.

5.1 Motivation

As noted in Section 4.2.4, the ability to control the communication of agents is generally not a feature of existing Multi-Agent System architectures but at the same time a central feature of our approach for agent-based Privacy-Preserving Information Filtering.

Anonymous communication is required for several interactions in our approach mainly in order to achieve unlinkability of user-related data. Depending on the actual scenario, solutions for sender anonymity as well as receiver anonymity are required.

We therefore provide the respective functionality within the Infrastructure Module, as specified in the following sections.

5.2 Analysis

This section describes the ontologies, roles and interactions of the Infrastructure Module. For the sake of readability, all tables and diagrams containing the formal specification may be found in Appendix A.1. Usually the analysis phase and thus the specification of interactions abstracts from agents and platform configurations. However, in this case it is necessary to refer to these concepts because of the reflective nature of the interactions.

When utilizing this module, it should be noted that the concepts of communication control and anonymous communication are mutually exclusive, because agents on a controlled platform are always identifiable when communicating, and thus cannot communicate anonymously.

The functionality required for controlling communication cannot be realized based on regular agent services and/or components, because an agent on a platform is usually not allowed to interfere with the actions of other agents in any way. Otherwise, the security of agents would be severely compromised. Therefore, additional infrastructure providing the required functionality has to be added to the MAS architecture itself.

Controlling the communication capabilities of an agent is realized by restricting its incoming and outgoing communication channels to specific platforms or agents on external platforms as well as other possible communication channels, such as the file system. These restrictions are stated via rules, similar to rules used by a firewall, which become effective only if the respective agent consents. Consent is required because otherwise the overall security would be compromised, as attackers could easily block various communication channels. Our approach does not require controlling the communication between agents on the same platform, and therefore this aspect is not addressed[1]. Consequently, all rules addressing communication capabilities have to be enforced across entire platforms, because otherwise a controlled agent could simply use a non-controlled agent on the same platform as a relay for communicating with agents residing on external platforms.

Obviously, an agent on a controlled platform should not be able to migrate freely to other platforms, because in that case control could be lost. Because migration is initiated by communicating with a remote platform manager agent, it is made impossible implicitly with the following non-critical exception: An agent on a controlled platform may still be able to migrate to

[1]While also depending on the actual MAS architecture utilized, this would typically be more complicated or even impossible to realize, because intra-platform agent communication is usually based on channels that are more difficult to control than inter-platform communication channels, such as communication within a single Java Virtual Machine as opposed to TCP/IP-based communication.

platforms it is allowed to communicate with.

Agents attempting to migrate onto a controlled platform should be made aware of this fact, or incoming migration should be prohibited altogether. The same applies to agents attempting to create additional agents on a controlled platform. Because these aspects are not directly relevant for the scenarios discussed in this work, we will not address them further.

The functionality for anonymous communication may either be realized based on regular agent services and/or components, or by adding additional infrastructure providing the required functionality to the MAS architecture itself (see Section 3.3.1 for related work in this area). In any case, some kind of relay is required because sender and receiver (i.e. agent service user and provider) cannot communicate directly without compromising anonymity.

5.2.1 Ontologies

Rules for controlling communication are expressed via the ontology "Communication Rules", for which see Section A.1 in the appendix. Basically, a rule specifies a controller (i.e. the controlling agent itself), a sender (i.e. the platform that is controlled) and receivers (a set of agents and/or platforms to be excepted from communication blocking).

For every controlled platform, a set of *activated rules* contains the rules which are applicable to the respective platform. Activated rules may contain different controlling agents. From the set of activated rules, which may be contradictory, a single *effective rule* is generated consisting of two parts. The first part is applied in order to decide which communication attempts to block, while the second part is applied in order to decide which platforms an agent on a controlled platform may control in turn. Generally, the effective rule is determined by creating the intersection of the sets of exceptions of each single activated rule. As an example, if the first activated rule states that communication with all platforms except P_1 and P_2 is to be blocked, and the second activated rule states that communication with all platforms except P_2 and P_3 is to be blocked, the effective rule in this case states that communication with all platforms except P_2 is to be blocked. All rules originating from agents on a specific platform are collected as a set of *foreign rules*.

Because a controlling agent is expected to intend to communicate with agents on the controlled platform, it is excepted from communication blocking in the first part of the effective rule. In cases where two or more different agents control one platform, only the activated rules related to the first agent are considered when determining the first part of the effective rule in order to ensure that the first controller is able to communicate with the controlled platform. For the second part of the effective rule, however, all activated rules are considered[2].

All information related to sender and receiver anonymity is expressed via the ontology “Anonymity”, for which see Section A.1 in the appendix. Sender anonymity requires a pseudonym to be used for the sender, the agent address of the actual interaction partner, information about the interactions that are to be anonymized, and information related to the required degree of unlinkability, i.e. whether all single interaction steps should be unlinkable. Receiver anonymity requires a pseudonym to be used for the receiver, the actual agent address of the receiver, and information about the interactions that are to be anonymized. Depending on the implementation of the anonymizer, it may be infeasible to provide a mechanism for continuous receiver anonymity, mainly because there may be a large number of potential receivers for each actual interaction which would have to reachable continuously. Therefore, a time slot may be given optionally indicating the time periods in which the anonymizer should actually facilitate anonymous interaction. In both cases, an optional attribute may be used to indicate the required multiplicity, i.e. whether the respective interaction should be carried out anonymously once, a fixed number of times, or an unlimited number of times.

5.2.2 Roles and Interactions

This section describes the roles and interactions of the Infrastructure Module. For the role schemas, see Appendix A.1. Table 5.1 provides an overview of the roles.

5.2.2.1 Communication Control

A role with special privileges exceeding those of regular roles, namely the SupervisorRole, is required for actually enforcing control of the communication capabilities of specific agents. On every platform hosting agents which are potential candidates for controlled agents, there has to be an agent realizing this role. Similar to the agent realizing the PlatformManagerRole itself, the agent realizing the SupervisorRole has to be trusted to act non-maliciously, i.e. to carry out all tasks as specified without trying to obtain additional information. In other words, these agent have to be part of the trusted environment described in Section 4.2.1.

To simplify matters, both roles may be realized by one single agent. We model controlling agents in the interactions described below as agents real-

[2]This is done in order to avoid subsequent complications in more complex situations, as described in Appendix B, Example 4.

Table 5.1: The roles participating in the Infrastructure Module.

	short name/ aggregated by		
role name	user	provider	filter
SupervisorRole	SR(U)	SR(P)	
Responsible for actually controlling the communication capabilities of agents realizing the ControllableAgentRole, and for handling request for control regarding agents on remote platforms.			
ControllableAgentRole	CAR(U)	CAR(P)	
Provides functionality that allows a ControllerRole to obtain the consent to control the agent realizing this role under certain circumstances.			
AnonymizerRole	AR(U)	AR(P)	
Provides functionality for anonymous communication.			
AgentRole	*(U)	*(P)	
Does not provide specific functionality. Participates in interactions with other roles.			

izing a generic AgentRole, because they do not have to provide specific functionality, and controlled agents as agents aggregating the ControllableAgentRole. The agent realizing the PlatformManagerRole and the agent realizing the SupervisorRole itself are always exempt from communication blocking, because they have to communicate with other platforms in order to carry out their respective tasks.

For clarity, we point out that the term “controlling agent” always refers to the agent that initiated the control, while the agent realizing the SupervisorRole is responsible for actually enforcing control.

To keep the complexity of the process of adding and handling rules manageable, a group of platforms is always blocked uniformly, i.e. each agent on any platform within the group may communicate with any other agent located on a platform within the group, with the controlling agent unless otherwise restricted, but with no one else. Therefore, a controlling agent may only to specify a group of platforms he intends to block, without being able to add additional restrictions, which would only complicate matters unnecessarily. Consequently, agents on a controlled platform are blocked from communication with any agent outside the respective group of controlled platforms, with the exception of the controlling agent itself.

Basic Interactions To facilitate controlling the communication of agents, the four basic interactions RestrictCommunication, CheckRule, ActivateRule, and AcquireConsent are provided as specified in Table A.1, Table A.2, Table A.3, and Table A.4 of the appendix. The partial use case "restrict communication" representing a typical scenario based on these interactions is shown in Figure 5.1. It should be noted that if consent is withheld by at least one role, further interactions are not carried out, and no rules are added or activated anywhere.

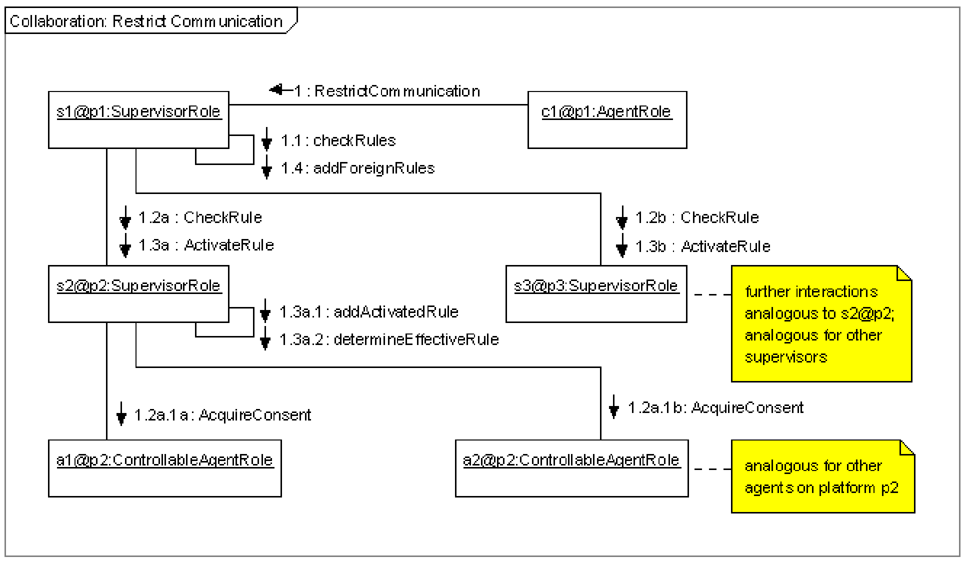

Collaboration: Restrict Communication
1 : RestrictCommunication
s1@p1:SupervisorRole
c1@p1:AgentRole
1.1 : checkRules
1.4 : addForeignRules
1.2a : CheckRule
1.3a : ActivateRule
1.2b : CheckRule
1.3b : ActivateRule
s2@p2:SupervisorRole
s3@p3:SupervisorRole
further interactions analogous to s2@p2; analogous for other supervisors
1.3a.1 : addActivatedRule
1.3a.2 : determineEffectiveRule
1.2a.1 a : AcquireConsent
1.2a.1 b: AcquireConsent
a1@p2:ControllableAgentRole
a2@p2:ControllableAgentRole
analogous for other agents on platform p2

Figure 5.1: Collaboration Diagram for the partial use case "restrict communication".

It may appear unnecessary to involve the supervisor, which is the agent realizing the SUPERVISORROLE, on the controlling agents's own platform in the process, instead of allowing the respective role to directly interact with the supervisor on the platform to be controlled. This direct interaction is not allowed because it would not allow scenarios based on cascading control, as described below. Furthermore, an agent's own supervisor checks the second part of its effective rule in order to determine whether the agent may control the remote platform as intended, and it checks for attempts to block the same group of platforms more than once by checking its foreign rules. Thus, potentially unsuccessful attempts to restrict communication are detected without unnecessary inter-platform interactions. A detailed example highlighting the use of these basic interactions, and the effect of multiple activated rules on the effective rule is given in Appendix B.

Revoking Control Revoking control by removing rules that have already been activated is required for several reasons: The tasks of controlling a group of platforms may have to be transferred from one agent to another, or a group of controlled platforms may have to be enlarged or downsized. Furthermore, a rule may have been activated as a precautionary act that subsequently turns out to be unnecessary, or a platform may have to be controlled temporarily only, e.g. because sensitive information handled by an agent on the platform has expired. Therefore, the interactions RevokeControl and RevokeRule are provided as specified in Table A.5 and A.6 of the appendix. The partial use case "revoke control" representing a typical scenario based on these interactions is shown in Figure 5.2 . When revoking control, the effective rules are determined based on the remaining active rules. The agents on the controlled platform are not required to consent. Only the controlling agent itself may revoke control. Again, Appendix B illustrates these interactions via an example. Similar to restricting communication, control is always revoked with respect to a group of platforms. Therefore, it is not possible to revoke control for a single platform within a group of controlled platforms directly. Instead, communication has to be restricted explicitly for a new group consisting of the remaining platforms, and in a second step control may be revoked on the original group of platforms. For convenience, an additional interaction could combine both steps, but this interaction is not strictly required and thus optional.

Cascading Control A scenario not addressed by the interactions introduced so far is the cascading control scenario in which an agent intends to control a platform containing agents which in turn control further platforms. In this case, these further platforms are added to the groups of platforms to be controlled, by returning the respective information as an additional output of the CheckRule interaction. The rules regarding these additional platforms are activated without acquiring consent of the respective agents, because they have already consented to be controlled. Instead, these agents are notified of the cascading control via the interaction InformAboutCascadingControl specified in Table A.7 of the appendix, because this information may help agents to determine whether to carry out a specific protocol for which cascading control is required. The partial use case "restrict communication (cascading control)" representing a typical scenario based on cascading control is shown in Figure 5.3. The initiator of the interaction RESTRICTCOMMUNICATION is informed about all platforms that are controlled in addition to the group given as input. From this point onwards, no distinction is made between platforms controlled through cascading control and platforms con-

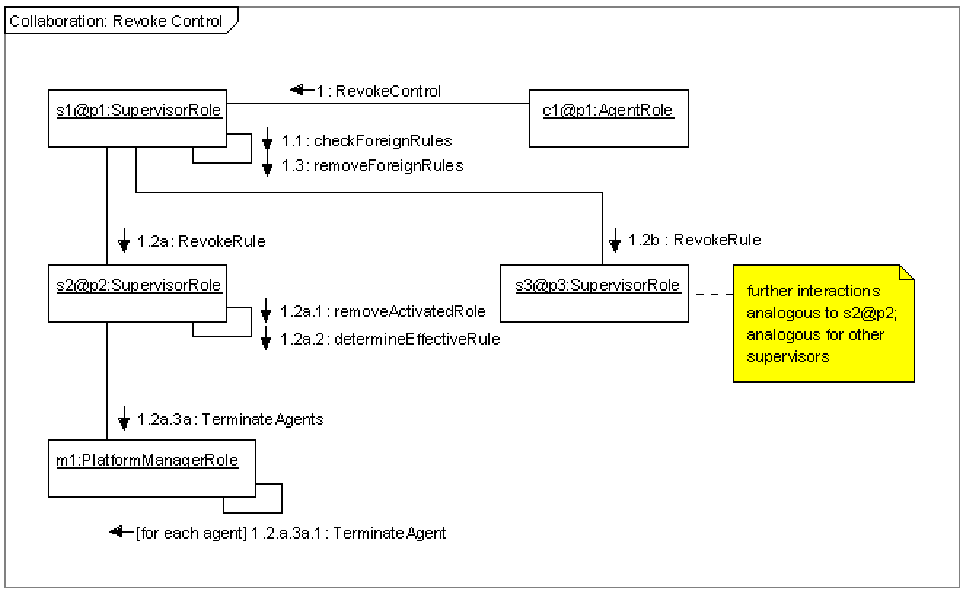
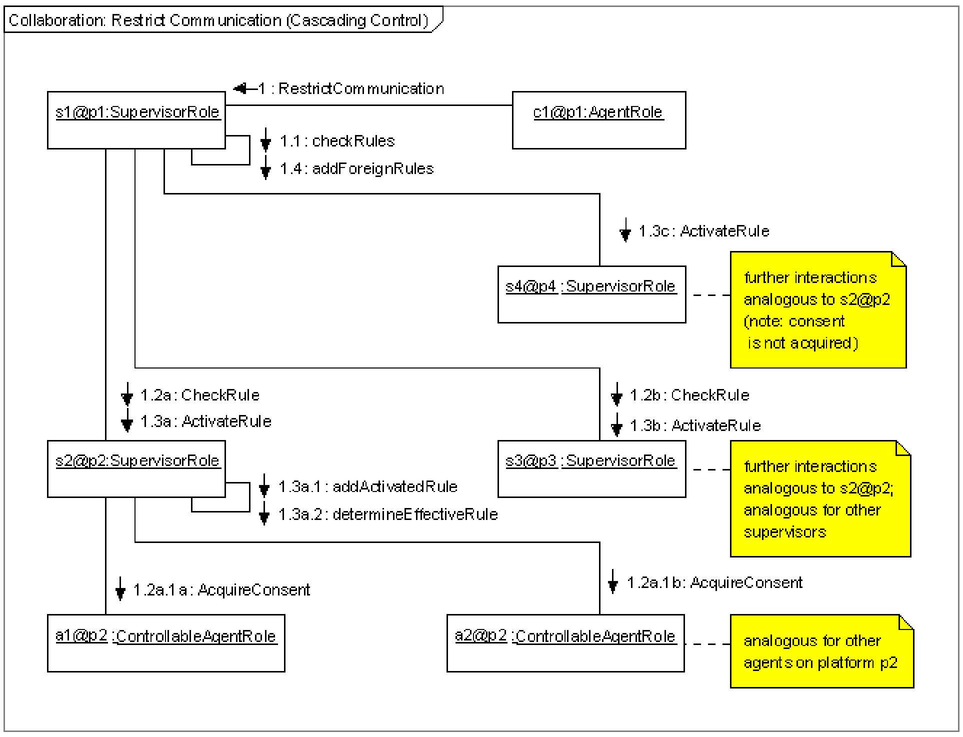

Collaboration: Revoke Control
1 : RevokeControl
s1@p1:SupervisorRole
c1@p1:AgentRole
1.1 : checkForeignRules
1.3 : removeForeignRules
1.2a: RevokeRule
1.2b : RevokeRule
s2@p2:SupervisorRole
s3@p3:SupervisorRole
further interactions analogous to s2@p2; analogous for other supervisors
1.2a.1 : removeActivatedRole
1.2a.2 : determineEffectiveRule
1.2a.3a : TerminateAgents
m1:PlatformManagerRole
[for each agent] 1.2.a.3a.1 : TerminateAgent

Figure 5.2: Collaboration Diagram for the partial use case "revoke control".

trolled regularly. Therefore, revoking control of a platform does not affect the activated rules regarding this platform even if they have been established through cascading control. Again, Appendix B provides an example for this scenario.

Additional Management Functionality As described above, controlling a certain agent is only possible by controlling an entire platform. Therefore, a large number of platforms is required in scenarios containing a large number of separate processes that have to be executed by agents controlled by different controlling agents. For example, in the Recommender System use cases of our PPIF approach, the optimal solution with regard to privacy and security is to use one separate platform for each process of the information filtering stage. For real-world applications aiming at a large number of users, the number of required platforms is therefore much larger than the usual number of platforms in a deployed system based on MAS architecture, which is typically relatively small due to the amount of resources required to run a platform.

For these reasons, it is infeasible to control at platform permanently, or in fact longer than for the time period that is sufficient for carrying out a small number of interactions. If control is short-lived, a controlled platform



Collaboration: Restrict Communication (Cascading Control)
1 : RestrictCommunication
s1@p1:SupervisorRole
c1@p1:AgentRole
1.1 : checkRules
1.4 : addForeignRules
1.3c : ActivateRule
s4@p4 :SupervisorRole
further interactions analogous to s2@p2 (note: consent is not acquired)
1.2a : CheckRule
1.3a : ActivateRule
1.2b : CheckRule
1.3b : ActivateRule
s2@p2:SupervisorRole
1.3a.1 : addActivatedRule
1.3a.2 : determineEffectiveRule
s3@p3 :SupervisorRole
further interactions analogous to s2@p2; analogous for other supervisors
1.2a.1 a : AcquireConsent
1.2a.1 b: AcquireConsent
a1@p2 :ControllableAgentRole
a2@p2 :ControllableAgentRole
analogous for other agents on platform p2

Figure 5.3: Collaboration Diagram for the use case “restrict communication (cascading control)”.

may be terminated or recycled and the respective resources may be re-used by subsequent for further tasks. An additional positive effect of short-lived control is that it further minimizes security risks. Short-lived control of a platform implies that the agents located on that platform are short-lived as well, because they will usually not be allowed to migrate away from a platform that is to be terminated, as this would result in a loss of control. Therefore, an agent consenting to be controlled implicitly consents to be terminated at any time as well.

The interactions for terminating controlled agents, RevokeControlAndTerminate and RevokeRuleAndTerminate, specified in Table A.8 and A.9 of the appendix, are very similar to the interactions defined above, with the additional result that the agents on the respective platforms are terminated either immediately after control is revoked or at a later time, depending on the following conditions: There must be no other remaining activated rule regarding the same controller, and the controller who started the service must be the first in the list of controllers. If the agents are not terminated immediately, an activated rule is added and evaluated whenever activated rules are removed via the interactions RevokeControl and RevokeControlAndTerminate, in order to terminate the agents at the appropriate point of time. Additionally, when a platform controlled by more than one agent is actually terminated, other supervisors are notified via the interaction InformAboutTermination specified in Table A.10 of the appendix. This interaction is also used in the cascading control scenario in order to inform the supervisors of platforms controlled by an agent to be terminated, in order to enable them to remove the respective activated rules as well. The partial use case "revoke control and terminate" representing a typical scenario based on these interactions is shown in Figure 5.4. Again, Appendix B illustrates this scenario. The interactions TerminateAgents and TerminateAgent are regarded as standard platform management interactions and therefore not specified further here.

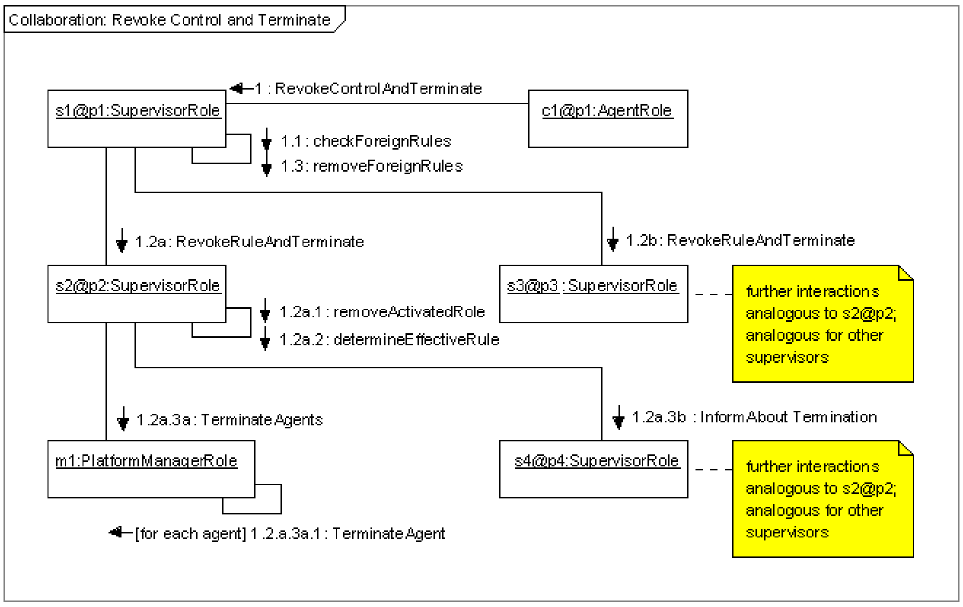

Collaboration: Revoke Control and Terminate
1 : RevokeControlAndTerminate
s1@p1:SupervisorRole
c1@p1:AgentRole
1.1 : checkForeignRules
1.3: removeForeignRules
1.2a: RevokeRuleAndTerminate
1.2b: RevokeRuleAndTerminate
s2@p2:SupervisorRole
s3@p3 :SupervisorRole
further interactions analogous to s2@p2; analogous for other supervisors
1.2a.1 : removeActivatedRole
1.2a.2: determineEffectiveRule
1.2a.3a: TerminateAgents
1.2a.3b : InformAboutTermination
m1:PlatformManagerRole
s4@p4:SupervisorRole
further interactions analogous to s2@p2; analogous for other supervisors
[for each agent] 1.2.a.3a.1 : TerminateAgent

Figure 5.4: Collaboration Diagram for the partial use case "revoke control and terminate".

Finally, an agent may actively request to be controlled by another agent, mainly in order to be able to carry out a protocol requiring this control at a certain point. The respective interaction RequestControl is provided as specified in Table A.11.

An aspect not addressed further in this work is the following: It may become necessary to enforce termination of platforms from a global platform manager's point of view, in cases where the controlling agents deliberately or accidentally fail to release control. This could be handled by using a time-out after which platforms are terminated automatically, or by using an incentive or billing mechanism that makes it desirable for a controlling agent to release control as soon as possible.

5.2.2.2 Anonymous Communication

The required functionality for anonymous communication is realized via an abstract AnonymizerRole. An agent requiring sender anonymity for a specific interaction relays that interaction through the AnonymizerRole, by interacting with it as if it where the actual interaction partner, i.e. the AnonymizerRole is the interaction partner in the interaction with the actual initiator, and the initiator in the interaction with the actual partner. The main interaction is preceded by the initiator providing the required information via the interaction SetupAnonymizer specified in Table A.12 of the appendix. The same interaction is used for realizing receiver anonymity, basically via the same relay mechanism. In this case, the agent requiring receiver anonymity is the partner in the main interaction, i.e. the roles of initiator and partner are reversed.

5.3 Design & Implementation

The design of the functionality for controlling communication and for anonymous communication is rather straightforward, because roles are mapped directly to agents and interactions to agent services as shown in Table 5.2 and Table 5.3 respectively, the only exception being the interactions CheckRule and ActivateRule, which are realized within a single agent service, because they are always used in conjunction. The implementation of agents, agent services and internal components is similarly straightforward and therefore omitted here. The remaining issue that has to be addressed with regard to communication control is the implementation of the actual control mechanism. As JIAC IV is based on Java, we utilize methods provided via the Java Security Manager as part of the Java security model. Thus, the supervisor agent is enabled to define custom security policies, thereby granting or denying other agents, which are executed as threads in the JVM, access to resources such as files or sockets for TCP/IP-based communication. Apart from denial of service attacks in which agents may attempt to disrupt the execution of other agents on the same platform by seizing a large amount of the available resources, all other malicious actions and communication attempts may be blocked via this mechanism.

Table 5.2: The mapping of interactions to agent services.

Interaction	Table	Agent Service
RestrictCommunication	A.1	*RestrictCommunication*
CheckRule ActivateRule	A.2 A.3	*ImplementRule*
AcquireConsent	A.4	*AcquireConsent*
RevokeControl	A.5	*RevokeControl*
RevokeRule	A.6	*RevokeRule*
InformAboutCascadingControl	A.7	*InformAboutCascadingControl*
RevokeControlAndTerminate	A.8	*RevokeControlAndTerminate*
RevokeRuleAndTerminate	A.9	*RevokeRuleAndTerminate*
InformAboutTermination	A.10	*InformAboutTermination*
RequestControl	A.11	*RequestControl*
SetupAnonymizer	A.12	*SetupAnonymizer*

Table 5.3: The mapping of roles to agents.

Role	Agent
AgentRole ControllableAgentRole	unspecified unspecified
SupervisorRole	**SupervisorAgent**
AnonymizerRole	**AnonymizerAgent**

As noted above, the **SupervisorAgent** and the agent realizing the PlatformManagerRole have to be part of a trusted environment, which may be realized e.g. based on a trusted computing infrastructure. As part of the deployment phase, however, the trusted environment does not affect the implementation phase and is therefore not discussed further at this point.

The remaining issue that has to be addressed with regard to anonymous communication is the implementation of the anonymizer itself. While theoretically any approach used for anonymous communication on the Internet may be mapped to a MAS context, resulting in an agent-based mix network, onion routing approach or a similar system (see Section 3.1.1 and Section 3.3.1 for related work in this area), we have chosen a simple proxy mechanism mainly because details of the anonymizer are out of the scope of this work.

The anonymizer therefore is implemented as a simple component, which may actually be part of the agent that intends to communicate anonymously, instead of a separate agent. The anonymizer component creates and terminates the actual relay agents as requested. The relay agents offer and execute the services that are to be used anonymously. Because they are deployed and controlled by the agent that intends to communicate anonymously, they are trusted implicitly, and a trusted third party is not required. Currently, the code for the relay agents has to be created manually, based on the respective service to be relayed. It is conceivable, however, to automatize this process and create the respective code dynamically at least for services without a user protocol.

5.4 Summary

This chapter describes basic functionality for controlling the communication capabilities of agents, and functionality for anonymous communication of agents because these kinds of functionality are generally not a feature of existing Multi-Agent System architectures but at the same time a central requirement of our approach for agent-based Privacy-Preserving Information Filtering.

We motivate the need for communication control and anonymous communication as additional functionality in MAS architectures for PPIF (Section 5.1). We specify ontologies containing the basic concepts, namely rules for communication control, sender anonymity, and receiver anonymity (Section 5.2.1). Regarding communication control, we specify roles and basic interactions for establishing and revoking control, interactions for cascading control, and interactions for additional management functionality (Section 5.2.2.1). Regarding anonymous communication, we specify a role and an abstract interaction (Section 5.2.2.2). Regarding the design and implementation of the specified functionality, we list agents and agent services, and we discuss the implementation of the actual control mechanism and the implementation of the anonymizer (Section 5.3). In the appendix, we give several examples illustrating the aspects of communication control (Appendix B).

The modules realizing the main use cases of our approach are based on functionality described in this chapter as well as functionality related to accessing persistent information in a well-defined manner, which is described in the following chapter.

Chapter 6

Transparent Persistence

This chapter describes an approach for Transparent Persistence in Multi-Agent Systems (TPMAS). It should be noted that it is possible to use this approach in any scenario involving the use of large amounts of persistent data, as it is not adapted especially for Privacy-Preserving Information Filtering. As motivated below, however, our approach for PPIF strongly benefits from a transparent persistence management mechanism, and therefore we include it here as a main component of the approach.

The chapter is structured as follows: Section 6.1 motivates the TPMAS Module. Section 6.2 describes the ontologies, roles and interactions of the module, while Section 6.3 describes the agents and agent services realizing these interactions. Section 6.4 concludes the chapter with a summary.

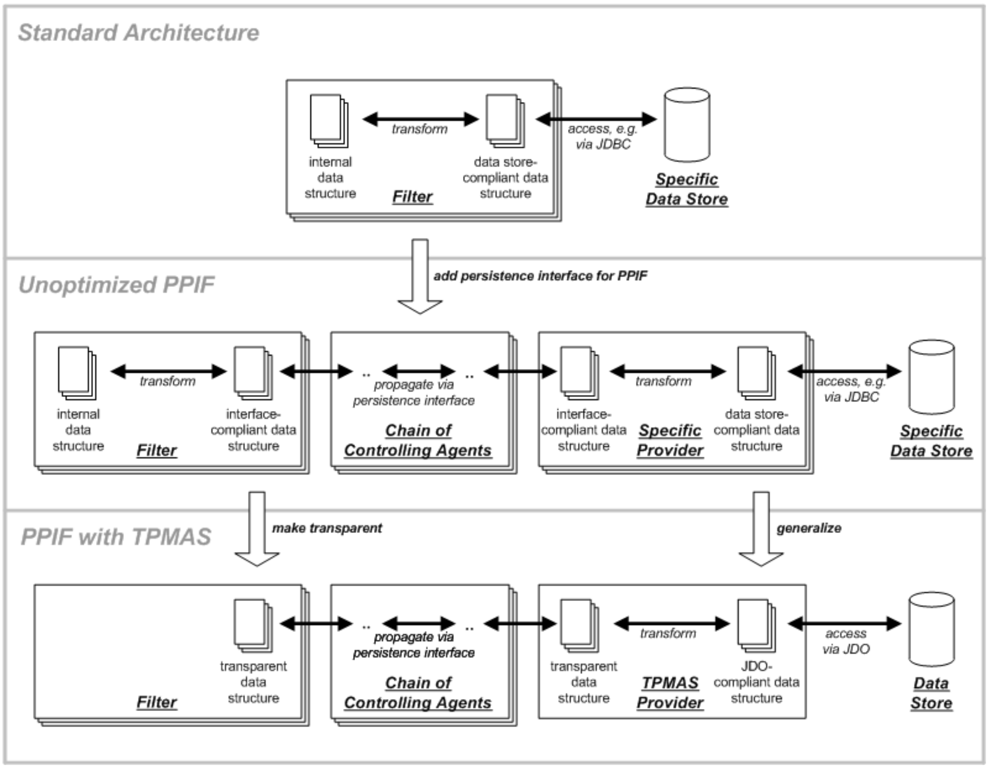
6.1 Motivation

Information Filtering in general is based on data that has to be stored persistently: It deals with users' long-term information needs, and therefore the respective user profile data has to be stored persistently. As in Information Retrieval, the provider data usually consists of large data sets that are relatively static[1], and is therefore best managed by using a persistent storage mechanism as well.

In standard Information Filtering architectures, the filter entity is not realized as independent from the provider entity. Therefore, the provider may use a specific persistent storage mechanism, which is usually a Relational Database Management System (RDBMS), and may utilize filtering techniques operating directly, e.g. via JDBC, on the data store. This straightfor-

[1]Static in the sense that while single items are usually added and removed constantly, the bulk of the provider data is kept permanently for a comparatively long time.

ward approach has the advantage that it may be optimized for performance easily. It is visualized in Figure 6.1 as "Standard Architecture".



Standard Architecture
transform
internal data structure
data store-compliant data structure
Filter
access, e.g. via JDBC
Specific Data Store
add persistence interface for PPIF
Unoptimized PPIF
transform
internal data structure
interface-compliant data structure
Filter
propagate via persistence interface
Chain of Controlling Agents
transform
interface-compliant data structure
Specific Provider
data store-compliant data structure
access, e.g. via JDBC
Specific Data Store
make transparent
generalize
PPIF with TPMAS
transparent data structure
Filter
propagate via persistence interface
Chain of Controlling Agents
transform
transparent data structure
TPMAS Provider
JDO-compliant data structure
access via JDO
Data Store

Figure 6.1: The motivation for a transparent and generic persistence mechanism in the context of PPIF. The topmost layer visualizes the data flow in a standard IF architecture. The layer below visualizes the additional operations introduced by our PPIF approach, and the bottommost layer shows subsequent optimizations leading to our final approach.

6.1.1 Persistence Interface

In our approach for Privacy-Preserving Information Filtering, the filter entity is actually independent from the provider entity. Therefore, the respective filter role cannot operate directly on the provider's data store, and an interface for accessing persistent information is required. Apart from functionality for retrieving profile data during the information filtering stage (i.e. read-only access to the persistent data), the interface should also provide functionality for storing and retrieving data to facilitate creating and updating profile models during the information collection stage and the information processing stage.

Though these models are dependent on the specific filtering technique, they cannot be stored internally by the filter role, because the temporary filter entities accessing these models are short-lived. Therefore, the profile models have to be stored by the user and provider role respectively.

Because the structure of a profile model is entirely dependent on the specific filtering technique, and filtering techniques may be based on arbitrary structures, it is not advisable to realize a persistence interface by providing various dedicated interaction for specific model structures (e.g. interactions with the goal of training a neural network, updating a decision tree or carrying out a clustering algorithm), because these interactions would always be potentially incomplete. Instead, the persistence interface should provide generic functionality for storing and removing data. Because the profiles and profile models, especially those on the provider side, are usually rather large, they should not have to be dealt with as a whole. Therefore, the persistence interface should also provide services for retrieving, storing and updating parts of a profile. The resulting data flow is visualized in Figure 6.1 as "Unoptimized PPIF".

6.1.2 Generic Transparent Persistence

In the Privacy-Preserving Information Filtering approach, the retrieval of potentially sensitive information is monitored by controlling the communication of agents. As an example, when retrieving parts of the provider profile within a filtering process, the controlling agent associated with the user role has to make sure no sensitive information about the user profile is used within the respective query. Therefore, a uniform query structure should be used regardless of the actual persistent storage mechanism (which may be a RDBMS, a file system, or something else). Otherwise, the controlling role would have to be adjusted to every single storage mechanism used.

While for the controller any uniform structure would be acceptable, a *transparent* persistence interface is preferable for the filter: It should be possible to persist the objects handled by the filter, and to use these objects when creating queries, rather than having to transform these objects into some other uniform structure within the filter. Otherwise, the mapping process would become needlessly complicated, as both the filter and the provider would have to carry out transformations. In the case of transparent persistence, transformations only have to be carried out by the provider. This aspect of transparent persistence is visualized in Figure 6.1 as the part of the final architecture "PPIF with TPMAS" related to the filter.

Finally, it should be possible to exchange the actual persistent storage mechanism without having to adjust any of the interactions. In the context of Object-Oriented Software Engineering[2], this is achieved via using a persistence mechanism such as the Java Data Objects (JDO) specification [65]. This aspect of generic persistence is visualized in Figure 6.1 as the part of the final architecture "PPIF with TPMAS" related to the provider.

6.2 Analysis

This section describes the ontologies, roles and interactions of the TPMAS Module. For the sake of readability, all tables and diagrams specifying these components may be found in Appendix A.2.

6.2.1 Ontologies

We do not use a special category for persistent objects, rather, it should be possible to treat objects of all categories defined in arbitrary ontologies as persistent objects. Therefore, ontologies defining certain categories may be used without need for adjustments. For reasons of access management and in order to keep a large number of persistent objects manageable, persistent objects are stored in groups, namely *contexts*. Every operation is applied to a single context, rather than globally to all persistent objects[3]. A context is referred to via a unique identifier.

Access control of contexts is supported by a simple authorization approach in which each context is assigned three authorization tokens for various access rights (read only access, read/write access, and full access including the right to create and terminate a context). These tokens may be propagated at the discretion of the agent that has created the respective context. More complex access control mechanisms, such as Role-Based Access Control, are not strictly required for our approach and are therefore not described here. If actually required, they may be added easily on top of the existing mechanism.

Apart from the categories required to create complex queries, which are collected in a separate ontology described below, only a few other basic categories are required for storing and retrieving persistent objects. For the respective ontology "Transparent Persistence", see Section A.2 of the appendix.

[2]Object-Oriented Software Engineering concepts are actually applicable in this case within the larger context of Agent-Oriented Software Engineering because the interactions between the provider agent and the data storage are not agent-based, and the internal functionality of agents is realized in an object-oriented manner.

[3]The analogous element in a database management system is a single database.

When retrieving objects from a context containing a potentially large number of objects, it is advisable to keep the size of the list of returned objects as small as possible, instead of retrieving a large list of perhaps only partially relevant objects. Therefore, the structure of the query construct used within the respective interaction should support complex queries, i.e. it should be possible to express queries in a manner similar to other query languages, such as SQL or JDOQL, the query language used in the JDO specification. For this purpose, an ontology-based query structure is provided which allows conjunctive and disjunctive queries on all attributes of the objects stored within a context. For the respective ontology "Query Construct", see again Section A.2 of the appendix.

6.2.2 Roles & Interactions

This section describes the roles and interactions of the TPMAS Module. For the role schemas, see Appendix A.2. Table 6.1 provides an overview of the roles. The only role actually specified in the TPMAS approach is the role providing, via transparent persistence, access to the actual persistent storage mechanism, namely the TPMASProviderRole. Because any role may use the services offered by this role, a designated service user role does not have to be specified and we use a generic AgentRole for the specification.

Table 6.1: The roles participating in the TPMAS Module.

	short name/ aggregated by		
role name	user	provider	filter
TPMASProviderRole	TPMAS(U)	TPMAS(P)	
Provides transparent access to a persistent storage mechanism.			
AgentRole	*(U)	*(P)	
Does not provide specific functionality. Participates in interactions with other roles.			

Two kinds of interactions are provided: Interactions operating on the context level, and interactions operating on the object level. Within the first group, the interactions CreateContext and TerminateContext for creating and terminating contexts are provided as specified in Table A.16 and Table A.17 of the appendix. Within the second group, the interaction ModifyObjects for storing, updating and removing objects within a contexts is provided as well as the interaction RetrieveObjects for retrieving objects, based on a query, as specified in Table A.18 and Table A.19 of the appendix. Because each interaction only involves the TPMASProviderRole and a generic AgentRole, we omit collaboration diagrams here.

6.2.3 Internal Functionality

It does not make sense to entirely specify the internal functionality of the TPMASProviderRole at this point, because it largely depends on the actual MAS architecture on the one hand, and the actual persistent storage mechanisms on the other hand. However, as m different MAS architectures and n different persistent storage mechanisms would require $m \cdot n$ different solutions, it seems appropriate to reduce this number by introducing further functionality. Using a mechanism for transparent persistence, the number of different solutions is in fact reduced to m, i.e. one per actual MAS architecture.

We therefore utilize the Java Data Objects (JDO) specification [65][4] as the basis for the mechanism for transparent persistence. As many MAS architectures are based on Java, this choice is obvious and at the same time not too restrictive. There are various open source and commercial implementations of the JDO specification, which are interchangeable. Therefore, our approach is not limited to one specific JDO implementation, and we do not have to choose one at this point.

The JDO specification contains the following main components:

- *PersistenceCapable* interface: For Java objects that are to be made persistent, the respective classes have to implement this interface in order to provided the required functionality in the form of fields and methods. In most JDO implementations, the required code is added automatically by *enhancing* the respective classes.

- *PersistanceManager* interface: Persistence managers implementing this interface manage groups of persistent objects and provide functionality for adding and removing persistent objects via *transactions*.

- *Query* interface: JDO implementations provide, via this interface, mappings from queries expressed in JDOQL to queries expressed in the query language of the persistent storage mechanism (e.g. SQL).

For classes implementing the *PersistentCapable* interface, XML-based metadata files have be supplied specifying how a class is to be persisted,

[4]The specification was originally developed under the Java Community Process as Java Specification Request (JSR) 12, and released in 2002 as JDO 1.0. An extension to the JDO specification has been developed as JSR 243, and released in 2006 as JDO 2.0.

and which fields are to be made persistent. This information is used when storing persistent objects, and most JDO implementations also provide tools that generate, based on this information, the actual structures persistent objects are stored in (e.g. tables within a relational database).

There are other solutions for transparent persistence of Java objects which could be used alternatively to achieve transparent persistence within MAS architectures. They are, however, usually less generic with regard to the persistent storage mechanism (as an example, object-relational mapping tools, such as Hibernate [12] or the Java Persistence API as part of the Enterprise Java Beans (EJB) 3.0 specification [67] require a Relational Database Management System as the persistent storage mechanism). Therefore, we have chosen the JDO specification as the most suitable solution for our approach.

6.3 Design & Implementation

This section describes the agents and agent services of the TPMAS Module, as well as internal functionality.

6.3.1 Agents & Agent Services

The design of the functionality for transparent persistence is rather straightforward, because roles are mapped directly to agents and interactions to agent services as shown in Table 6.2 and Table 6.3 respectively. The implementation of agents, agent services and internal components is similarly straightforward and therefore omitted here.

Table 6.2: The mapping of interactions to agent services.

Interaction	Table	Agent Service
CreateContext	A.16	*CreateContext*
TerminateContext	A.17	*TerminateContext*
ModifyObjects	A.18	*ModifyObjects*
RetrieveObjects	A.19	*RetrieveObjects*

Table 6.3: The mapping of roles to agents.

Role	Agent
AgentRole	unspecified
TPMASProviderRole	**TPMASProviderAgent**

6.3.2 Internal Functionality

In order to facilitate the utilization of a JDO implementation within a MAS architecture, we provide the following functionality:

- Functionality for dynamically creating a Java class for a category of an ontology, and a bidirectional mapping of objects of this Java class to objects of the category. MAS architectures may use Java classes themselves as ontology categories, without using a separate language in which ontologies are expressed. In this case, the functionality obviously does not have to be actually implemented. However, other MAS architectures, such as JIAC IV, use a separate language for specifying ontologies. Therefore, in this case Java classes have to be created dynamically. A dynamically created Java class representing a category of an ontology contains fields matching the attributes of the category (thus, a mapping of the base types defined in the respective ontology language to Java base types is required), a constructor with the category attributes as input parameters, and, for the reverse mapping, methods with the category attributes as return parameters.

- Functionality for creating all required metadata information and files for a given category. Metadata information is normally not created automatically in order to give the developer greater control over the data storage schema and the way objects are made persistent. Because this course of action is usually not required in our approach, we actually create all metadata information automatically, which turns out to be a rather straightforward task. It should be noted, though, that ontology objects may also be mapped to already existing elements stored in a persistent storage mechanism. In this case, the respective metadata has to be created manually, based on the structure of the stored elements.

Figure 6.2 shows the main tasks carried out by our TPMAS implementation and a JDO implementation for handling persistent objects and queries, including all aspects of the previous list. Note that the tasks related to preparing a context, while shown separately from the tasks related to storing an object, are carried out, if necessary, immediately before storing an object and not as part of a separate agent service. Therefore, they are hidden from the service user who does not have to keep track of whether a context has already been prepared for storing objects of a certain category.

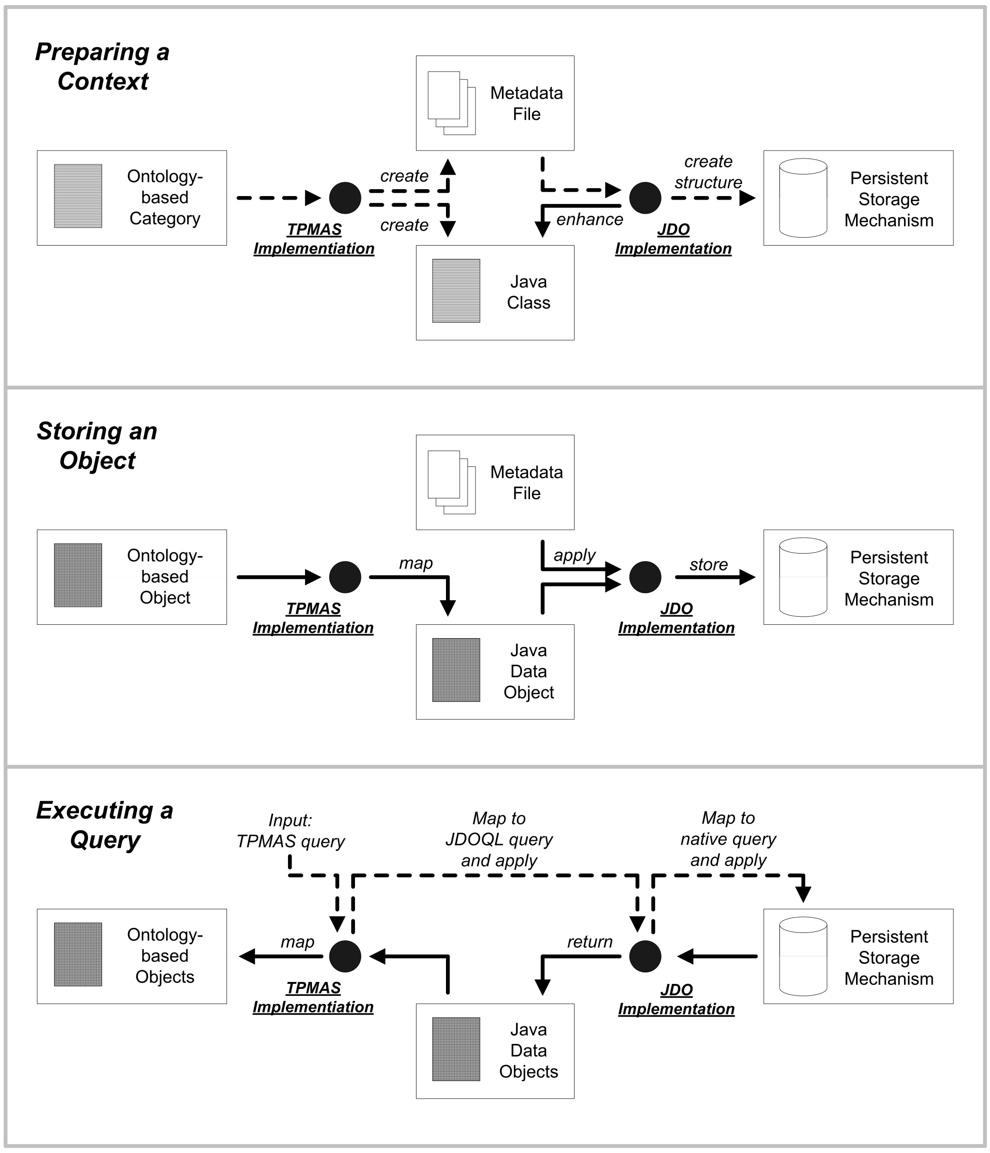


Figure 6.2: Overview of tasks provided by the TPMAS Module implementation and a JDO implementation for handling persistent objects and queries.

6.4 Summary

This chapter describes an approach for Transparent Persistence in Multi-Agent Systems (TPMAS) which is a main component of our approach for Privacy-Preserving Information Filtering, but at the same time may be used independently in any scenario involving the use of large amounts of persistent data.

We motivate the concept of transparent persistence by showing that our approach for PPIF requires a persistence interface and benefits from generic transparent persistence (Section 6.1). We specify ontologies containing the basic concepts (Section 6.2.1). We specify roles and basic interactions for handling contexts and objects (Section 6.2.2), and we discuss internal functionality required for our approach (Section 6.2.3). Regarding the design and implementation of the specified functionality, we list agents and agent services (Section 6.3.1), and we discuss the implementation of the internal functionality (Section 6.3.2).

With the functionality described in this and the previous chapter as a foundation, we are now able to describe the modules realizing the main use cases of our approach in the following chapters.

Chapter 7

The Recommender Module

This chapter describes functionality provided by the Recommender Module, i.e. it primarily addresses the use cases “get prediction for item” and “get recommendations”, as defined in Section 4.1. Moreover, it addresses the use cases related to the first two IF stages, namely the use cases “update profile elements” and “update profile model”. In addition to the use of the Recommender Module functionality in a Recommender System context, this chapter also covers its use in a Hybrid IF System context, because the interactions are largely similar in both cases.

The chapter is structured as follows: Section 7.1 briefly motivates the Recommender Module. Section 7.2 describes the ontologies, roles and interactions of the module, while Section 7.3 describes the agents and agent services realizing these interactions. Section 7.4 concludes the chapter with a summary.

7.1 Motivation

The Recommender Module constitutes one of the two core modules of our approach for Privacy-Preserving Information Filtering: Together with the Matchmaker Module, it addresses all use cases defined in Section 4.1. It provides primary functionality related to the requirements of user privacy, provider privacy and filter privacy (see Table 4.2). Thus, the need for functionality described in this chapter is motivated directly by the outline of our solution given in Section 4.2, as the abstract IF protocols introduced in the outline are realized via agent interactions, i.e. as part of agent services.

7.2 Analysis

This section describes the ontologies, roles and interactions of the Recommender Module. For the sake of readability, all tables and diagrams specifying these components may be found in Appendix A.3.

7.2.1 Ontologies

The main ontology of this module, namely the ontology "Information Filtering" shown in Figure A.5 of the appendix, contains categories and attributes that are directly derived from the definitions given in Section 2.2.1. These are explained further in the context of the interactions they are used in.

7.2.2 Roles and Interactions

This section describes the roles and interactions of the Recommender Module. For the role schemas, see Appendix A.3. The main abstract entities (user entity, provider entity, and filter entity) introduced in Section 2.2.1 are split into and mapped to different roles, each providing specific functionality as described in Table 7.1. The roles InterfaceRole, ProfileManagerRole, RelayRole, and TPMASProviderRole are aggregated by the user entity as well as by the provider entity. The roles TFERole and TFEFactoryRole are exclusively aggregated by the filter entity.

Interactions are defined according to the three stages of Information Filtering in the following sections. In these steps, all participating roles are assumed to act in an honest or at least honest-but-curious manner, i.e. they follow the specified protocols (see Section 2.3.3.1 for definitions of adversary models). Roles aggregated by the same abstract entity are assumed to always act in an honest manner with regard to each other, i.e. in interactions within the respective abstract entity. Additionally, these interactions are considered to be unobservable with regard to other roles aggregated by different entities[1]. Thus, threats related to privacy have to be considered whenever interactions between roles aggregated by different abstract entities take place. Additional threats emanating from roles acting in a malicious manner are discussed and addressed in Section 7.3.1. They do not have to be considered here because it turns out that they are addressable by refining the single protocol steps of interactions, from which we abstract in the analysis phase.

The TPMASProviderRole only interacts with the ProfileManagerRole of the respective abstract entity, via the interactions specified

[1]As discussed in Section 2.4.2, we consider this condition to be fulfilled in the underlying MAS architecture

Table 7.1: The roles participating in the Recommender Module.

	short name/ aggregated by		
role name	user	supplier	filter
InterfaceRole	IR(U)	IR(S)	
Responsible for interaction with human users or other software.			
ProfileManagerRole	PMR(U)	PMR(S)	
Responsible for the management of a profile, which may be accessed through this role only. Provides agents realizing the RelayRole. Responsible for controlling agents agents of other main abstract entities realizing the RelayRole.			
RelayRole	RR(U)	RR(S)	
Responsible for controlling agents of other main abstract entities realizing the RelayRole and the TFERole.			
TPMASProviderRole	TP(U)	TP(S)	
Provides transparent access to a persistent storage mechanism.			
TFERole			TFE
Carries out tasks of the information processing stage and the information filtering stage.			
TFEFactoryRole			FF
Provides agents realizing the TFERole.			

in Section 6.2.2. We omit this interactions, which may be mapped to the interactions of the ProfileManagerRole in a straightforward manner, in the following.

7.2.2.1 Information Collection Stage

The information collection stage deals with interactions related to creating and updating profiles. Because the basic profile data associated with a given abstract entity does not depend on a specific filtering technique, and is not directly related to a second entity, there are actually no threats related to privacy that have to be addressed. As an example, a human user may add his favorite movie to his personal profile via a Graphical User Interface (GUI) provided by the InterfaceRole of his personal agent in an unobservable way, i.e. even without the entity that originally provided the related information noticing. The user may subsequently remove the movie from his profile, or add movies from other sources in the same way. The information provider profile is updated in a similar manner, though usually on a larger scale and via an API rather than a GUI.

Therefore, this stage requires only basic interactions addressing the use case "update profile elements", namely the interactions UpdateProfile and QueryProfile , which provide functionality for updating and querying profiles as specified in Table A.21 and Table A.22 of the appendix. Figure 7.1 illustrates these interactions via the partial use case "create user profile" as a special case of the main use case "update profile elements".

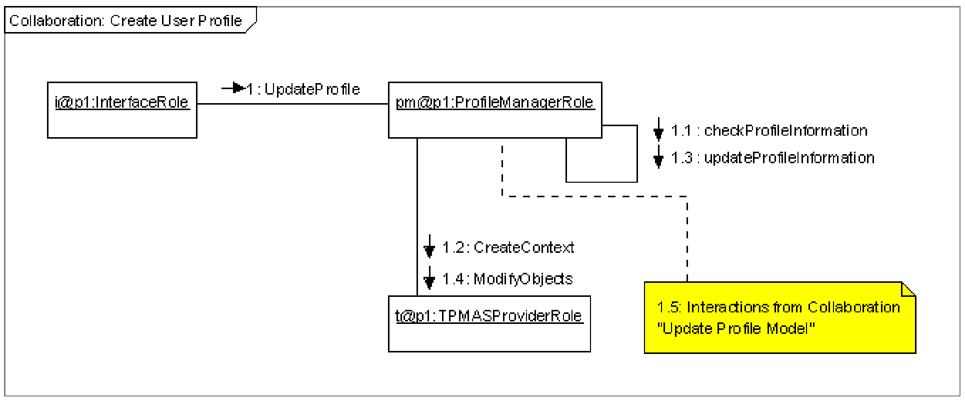

Collaboration: Create User Profile
i@p1:InterfaceRole
1 : UpdateProfile
pm@p1:ProfileManagerRole
1.1 : checkProfileInformation
1.3 : updateProfileInformation
1.2 : CreateContext
1.4 : ModifyObjects
t@p1:TPMASProviderRole
1.5: Interactions from Collaboration "Update Profile Model"

Figure 7.1: Collaboration Diagram for the partial use case "create user profile" as a special case of the main use case "update profile elements".

It may seem unnecessary to specify these additional interactions as we have specified similar interactions for transparent persistence in the previous chapter. These interactions are in fact used by the ProfileManagerRole in order to store profiles persistently. They are not used directly, however, because of additional functionality (for which see Section 7.2.2.2) provided by the ProfileManagerRole, which is partly triggered by the interactions of the Information Collection stage. Additionally, this course of action keeps the overall architecture flexible, because the relation between profiles and contexts is not fixed and may be arranged by the ProfileManagerRole as it sees fit[2], and it keeps the interface for the interaction partner simple because it does not have to deal with managing contexts and access control data.

[2]For example, a large profile may be stored across multiple databases and therefore in different contexts, which may even be managed by different agents implementing the TPMASProviderRole. These details are likely to be irrelevant for the respective service user.

7.2.2.2 Information Processing Stage

The raw profile data collected in the first stage may be used directly as input for a filtering technique in the Information Filtering stage. More complex filtering techniques, however, require a further processing of the collected data, resulting in models structuring the profile data in a certain way. Models may be used on both user profile and provider profile data, or only on data of a single profile, again depending on the filtering technique.

Different filtering techniques, such as minor variations of the same main technique, may use the same profile model. Therefore, the ontology "Information Filtering" groups filtering techniques by the profile model they are based on. In order to be able to create and maintain a model at this stage, the filtering technique to be applied in the following stage has to be known. If different filtering techniques are to be applied, different corresponding models have to be maintained. In principle, the required models could be created as a first step of the Information Filtering stage itself, but this approach is usually infeasible due to the complexity of the process combined with the fact that the Information Filtering stage may be initiated directly by a human user waiting for the results, which makes it more time-critical than the preceding stages.

Nevertheless, for all but the most basic models, the algorithm used to create and maintain the profile models should be considered part of the filtering technique itself and is therefore provided by the filter entity. Thus, two entities are involved in each process of the Information Processing stage, and privacy aspects have to be addressed[3]. The filter entity is responsible, via functionality provided by the a TFERole, for creating and updating the profile models. The agent realizing this role is located on a platform controlled by a user or a provider entity, depending on the profile on which a model is to be created or updated. Interactions specified in Section 5.2 are used for controlling the platform. The agent realizing the TFERole is created via a manager role, the TFEFactoryRole, via the interaction ObtainTFE specified in Table A.23 of the appendix. Figure 7.2 illustrates the respective partial use case "set up temporary filter entity".

Due to the complexity of the process, it is advisable to create and update large profile models independent of the actual information filtering process itself. Therefore, it is necessary for the respective abstract entity to announce an intended future use of a certain group of filtering techniques to its ProfileManagerRole in order to trigger the creation of the respective model.

[3]Note that while all three abstract entities are involved in collaboration-based processes of the Information Processing stage, we do not have to address this complication in the context of Recommender System functionality, for reasons discussed in Section 2.2.1.

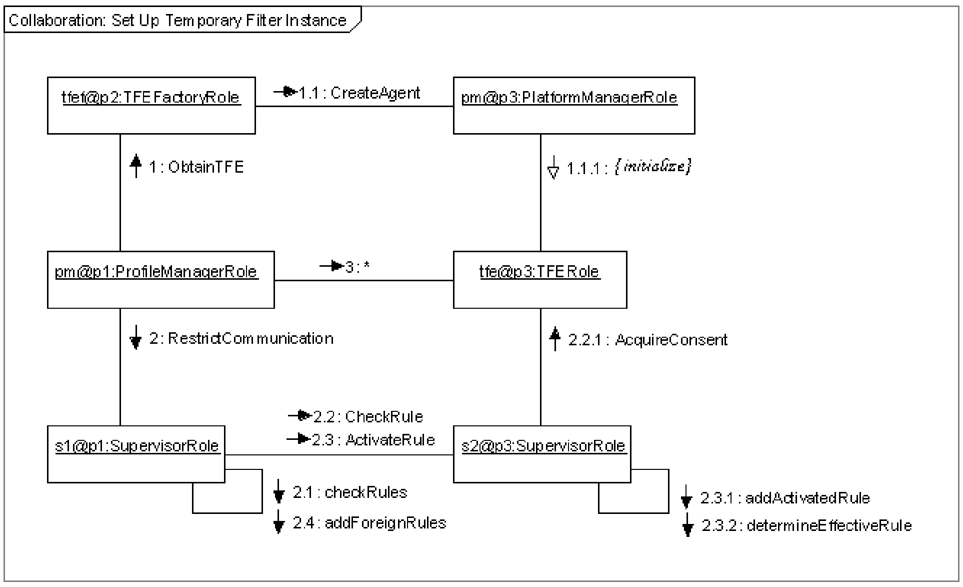


Figure 7.2: Collaboration Diagram for the partial use case "set up temporary filter entity".

For this reason the interaction SetUpdatePolicy specified in Table A.24 of the appendix is provided by the PROFILEMANAGERROLE. It allows its initiator to define a profile model update policy and a group of filtering techniques to be used on a the respective profile or group of profiles.

Additionally, the PROFILEMANAGERROLE acts as a relay between the TFEROLE and the TPMASPROVIDERROLE that handles the persistent storage of the profiles. Therefore, similar to the profile management interactions introduced in Section 7.2.2.1, the additional interactions UpdateProfileModel and QueryProfileModel are provided as specified in Table A.25 and Table A.26 of the appendix. These interactions are required at this stage because the TFEROLE realized by an agent on a controlled platform cannot communicate with the TPMASPROVIDERROLE directly (unless the controller agent itself realizes this role as well) and therefore a relay is required.

Finally, for creating and modifying a profile model, the interaction ModifyProfileModel is provided as specified in Table A.27 of the appendix. This interaction is initiated by the PROFILEMANAGERROLE, based on the respective update policy: If a profile is to be updated immediately, the interaction is triggered whenever the interaction UpdateProfile is carried out. If a profile is to be updated periodically, it is triggered by an internal timer. If an update policy implies that a new profile model has to be created, it is started immediately as well. Figure 7.3 illustrates the respective main use case "update profile model".

In each case, the ProfileManagerRole is responsible for supplying the appropriate profile elements. In the first case, they may be carried over directly from the respective profile management service. In the second case, the ProfileManagerRole either has to keep track of all elements received after the last profile model update or, as is done in the third case as well, use the interaction RetrieveObjects to obtain the appropriate elements.

These interactions are sufficient if the TFERole is assumed to be honest or at least honest-but-curious, because there is no way to propagate private information outside the specified interactions.

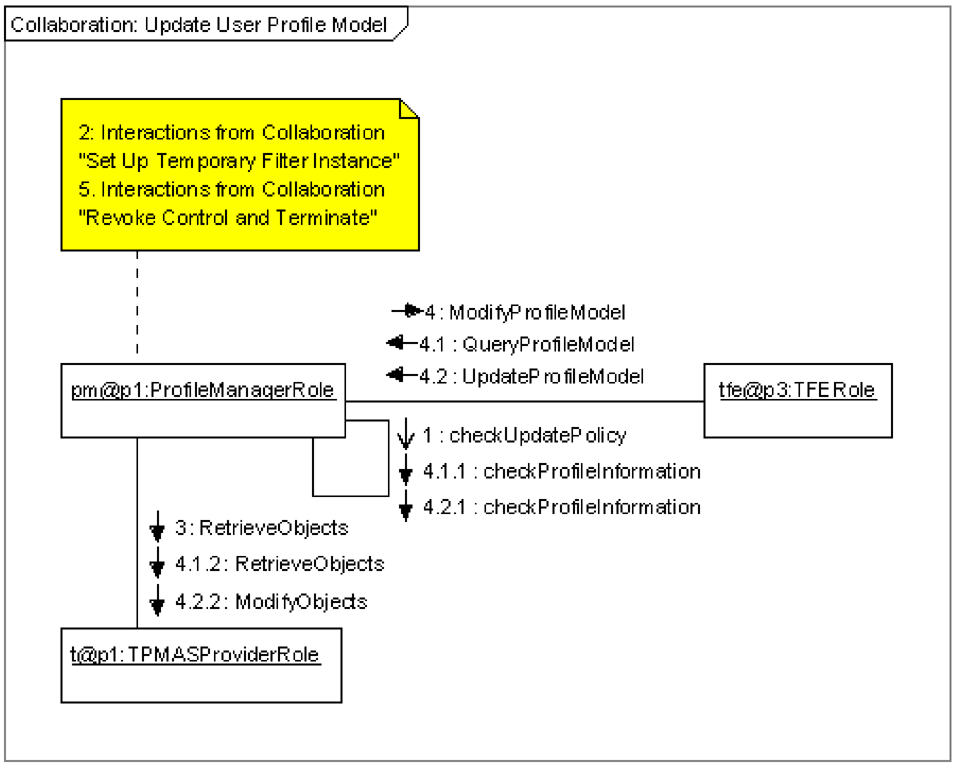

Collaboration: Update User Profile Model
2: Interactions from Collaboration "Set Up Temporary Filter Instance"
5. Interactions from Collaboration "Revoke Control and Terminate"
4 : ModifyProfileModel
4.1 : QueryProfileModel
4.2 : UpdateProfileModel
pm@p1:ProfileManagerRole
tfe@p3:TFERole
1 : checkUpdatePolicy
4.1.1 : checkProfileInformation
4.2.1 : checkProfileInformation
3: RetrieveObjects
4.1.2: RetrieveObjects
4.2.2: ModifyObjects
t@p1:TPMASProviderRole

Figure 7.3: Collaboration Diagram for the main use case "update profile model".

If the filter entity considers the generated model to contain sensitive information, such as data that could be analyzed in order to obtain information about the algorithm used by the respective filtering technique, the model should be regarded as private data of the filter entity and subsequently it should be protected accordingly. As described in more detail in the context of exemplary filtering techniques in Chapter 9, this may be achieved by encrypting the model before it is propagated from the filter entity to another entity.

7.2.2.3 Information Filtering Stage

The final stage providing the information filtering process itself uses the data collected and processed in the preceding two stages and compares two profiles in order to generate recommendations or a prediction for a given item. Three different abstract entities are involved in this stage: The user entity, a supplier entity which may be a provider entity (in a Recommender System context) or a different user entity (in a Hybrid IF System context[4]), and a filter entity. Therefore, it is the most complex stage with regard to privacy threats.

The TFERole introduced in the previous section is used in this stage to carry out the actual filtering process. It is neither required nor possible to actually use the same agent for both tasks, because the respective agent is terminated at the end of the information processing stage. With regard to functionality, the TFERole actually aggregates two partial roles, one used in the Information Processing stage and one used in the Information Filtering stage, because different algorithms may be used in these stages. However, the partial roles have to dovetail in order for the actual filtering technique to be applicable to the generated profile models. Apart from the actual algorithm applied, they are utilized in similar manners. Therefore, we subsume these partial roles as the TFERole.

Based on the outline of the information filtering process described described in Section 4.2, we describe the essential interaction steps for the use cases based on linkable result data and private result data (including the Hybrid IF System scenario) in Table 7.2 and Table 7.3 respectively.

For the Hybrid IF System scenario, we assume the result data to be completely private, mainly because there is no reason why the supplier, who in this case represents another user, should obtain the result data which is part of his user profile and as such no new information. Therefore, the protocol outlined in Table 7.3 could be used in this case. However, in order to provide an additional incentive for the supplier to participate in the process at all, additional result data should returned by the TFERole which is actually relevant for the supplier, such as recommendations taken from the user profile. While this could be achieved by applying the protocol with the roles of user and supplier reversed, it is easier and more efficient to propagate

[4]While this chapter focuses on Recommender System functionality, the introduced protocols may also be used in a Hybrid IF System, as described in Section 8.2.2.3. For this reason, we describe them in a generalized form here.

Table 7.2: The essential interaction steps of the information filtering stage for the use cases based on linkable result data, based on the abstract protocol shown in Figure 4.1. In the case of semi-linkable result data, the user entity roles have to remain anonymous in all interactions with roles of other entities.

Step	*Sender* → *Receiver*	Message	part of interaction
I.a	$RR(U)$ restricts communication of TFE		
I.b	$PMR(U) \rightarrow RR(U)$	PR_u	QueryProfileModel
I.c	$RR(U) \rightarrow TFE$	PR_u	QueryProfileModel
II.a	$PMR(S)$ restricts communication of $RR(U)$,TFE		
II.b	$PMR(S) \rightarrow RR(U)$	PR_s	QueryProfileModel
II.c	$RR(U) \rightarrow TFE$	PR_s	QueryProfileModel
III.a	$TFE \rightarrow RR(U)$	RES	GetResultsAsUser
III.b	$RR(U) \rightarrow PMR(S)$	RES	GetResultsAsSupplier
III.c	$PMR(S) \rightarrow PMR(U)$	RES	GetResults
III.d	$RR(U)$ terminates TFE		
III.e	$PMR(S)$ terminates $RR(U)$		

all result data in a single protocol. In this case, the result data contains specific information for the user and the supplier, i.e. $RES = RES_u \cup RES_s$.

Because the TFERole uses sensitive information related to two different abstract entities in this stage, it has to be controlled by both entities, or, more precisely, by each entity as soon as the respective sensitive information is provided. As described in Section 5.2, effective control by more than one controller can only be established through cascading control, i.e. one of the controllers has to be controlled in turn.

We therefore introduce an additional role, the RelayRole^{User}. This role, which is aggregated by the abstract user entity, controls the TFERole and is in turn controlled by the ProfileManagerRole^{Provider}. This second control is established after the TFERole has received all user profile data required for the filtering process. This is done for the following reason: If control of the RelayRole^{User} would be established at the beginning of the filtering process, the user profile information would have to be communicated via the ProfileManagerRole^{Supplier}, and the privacy of the user could not be preserved. Using this construction, however, there is no way for the TFERole to receive user profile data after a certain point, and especially not after having received supplier profile data. This limitation has to be taken into account when designing or selecting suitable filtering techniques, and is therefore addressed in Chapter 9. For use cases based on private result data, an additional RelayRole^{Supplier} is required for the result propagation,

Table 7.3: The essential interaction steps of the information filtering stage for the use cases based on private result data, based on the abstract protocol shown in Figure 4.2. In the case of semi-private result data, all user roles must remain anonymous in interactions with other roles.

Step	$Sender \rightarrow Receiver$	Message	part of interaction
I.a	$RR(U)$ restricts communication of TFE		
I.b	$PMR(U) \rightarrow RR(U)$	PR_u	QueryProfileModel
I.c	$RR(U) \rightarrow TFE$	PR_u	QueryProfileModel
II.a	$RR(S)$ restricts communication of $RR(U), TFE$		
II.b	$PMR(S) \rightarrow RR(S)$	PR_s	QueryProfileModel
II.c	$RR(S) \rightarrow RR(U)$	PR_s	QueryProfileModel
II.d	$RR(U) \rightarrow TFE$	PR_s	QueryProfileModel
III.a	$PMR(U)$ restricts communication of $RR(S), TFE$		
III.b	$TFE \rightarrow RR(U)$	RES	GetResultsAsUser
III.c	$RR(U) \rightarrow RR(S)$	RES	GetResultsAsSupplier
III.d	$RR(S) \rightarrow PMR(U)$	RES	GetResults
For semi-private result data:			
repeat 3.5 $\forall$ $res \in RES$:			
III.e	$PMR(U) \rightarrow PMR(S)$	res	ExchangeResults
For completely private result data:			
III.e	omitted		
For the Hybrid IF System scenario:			
III.e	$PMR(U) \rightarrow PMR(S)$	RES_s	ExchangeResults
III.f	$RR(U)$ terminates TFE		
III.g	$RR(S)$ terminates $RR(U)$		
III.h	$PMR(U)$ terminates $RR(S)$		

as described in Table 7.3.

The additional interactions required in this stage, namely the interactions GetResultsInternally, GetResults, GetResultsAsSupplier, GetResultsAsUser, ExchangeResults, and ObtainRelay are specified in Table A.28, Table A.29, Table A.30, Table A.31, Table A.32, and Table A.33 of the appendix. Finally, the interaction ShareKeys as specified in Table A.34 of the appendix is used by roles aggregated by the same abstract entity for exchanging keys used in encryption schemes. It is included here because while it is not required as long as only honest and honest-but-curious participants are assumed, is is required in case of malicious participants, as described in Section 7.3.1.

Figure 7.4 illustrates the use cases "get recommendations" and "get pre-

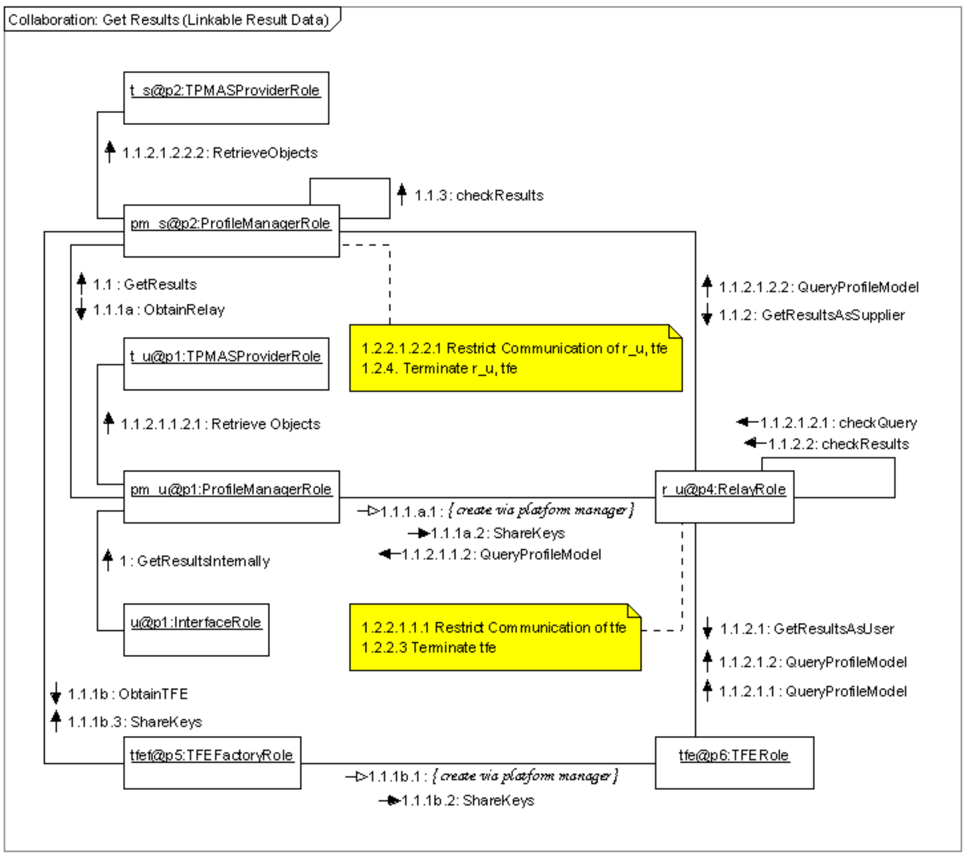

Collaboration: Get Results (Linkable Result Data)
t_s@p2:TPMASProviderRole
1.1.2.1.2.2.2: RetrieveObjects
1.1.3: checkResults
pm_s@p2:ProfileManagerRole
1.1 : GetResults
1.1.1a : ObtainRelay
1.1.2.1.2.2: QueryProfileModel
1.1.2: GetResultsAsSupplier
t_u@p1:TPMASProviderRole
1.2.2.1.2.2.1 Restrict Communication of r_u, tfe
1.2.4. Terminate r_u, tfe
1.1.2.1.1.2.1 : Retrieve Objects
1.1.2.1.2.1 : checkQuery
1.1.2.2: checkResults
pm_u@p1:ProfileManagerRole
r_u@p4:RelayRole
1.1.1.a.1 : {create via platform manager}
1.1.1a.2: ShareKeys
1.1.2.1.1.2: QueryProfileModel
1 : GetResultsInternally
u@p1:InterfaceRole
1.2.2.1.1.1 Restrict Communication of tfe
1.2.2.3 Terminate tfe
1.1.2.1: GetResultsAsUser
1.1.2.1.2: QueryProfileModel
1.1.2.1.1: QueryProfileModel
1.1.1b : ObtainTFE
1.1.1b.3: ShareKeys
tfef@p5:TFEFactoryRole
tfe@p6:TFERole
1.1.1b.1 : {create via platform manager}
1.1.1b.2: ShareKeys

Figure 7.4: Collaboration Diagram for the main use cases "get recommendations" and "get prediction for item", based on linkable result data in a Recommender System context.

diction for item", based on linkable result data in a Recommender System context. Figure 7.5 illustrates the same use cases, based on private result data in a Recommender System context.

Query Data Propagation As indicated by its name, the RelayRole^{User} has to act as a relay for interactions initiated by the TFERole. It participates in the interactions QueryProfile and QueryProfileModel as well, but instead of interacting directly with a TPMASProviderRole (as the ProfileManagerRole does), it relays the queries by interacting with the ProfileManagerRole of the other abstract entity.

In the Hybrid IF System scenario, where the supplier represents a different user entity, entire user profiles may be retrieved independent of each other because they are generally rather small. In the Recommender System scenario, however, retrieving an entire provider profile is often infeasible due

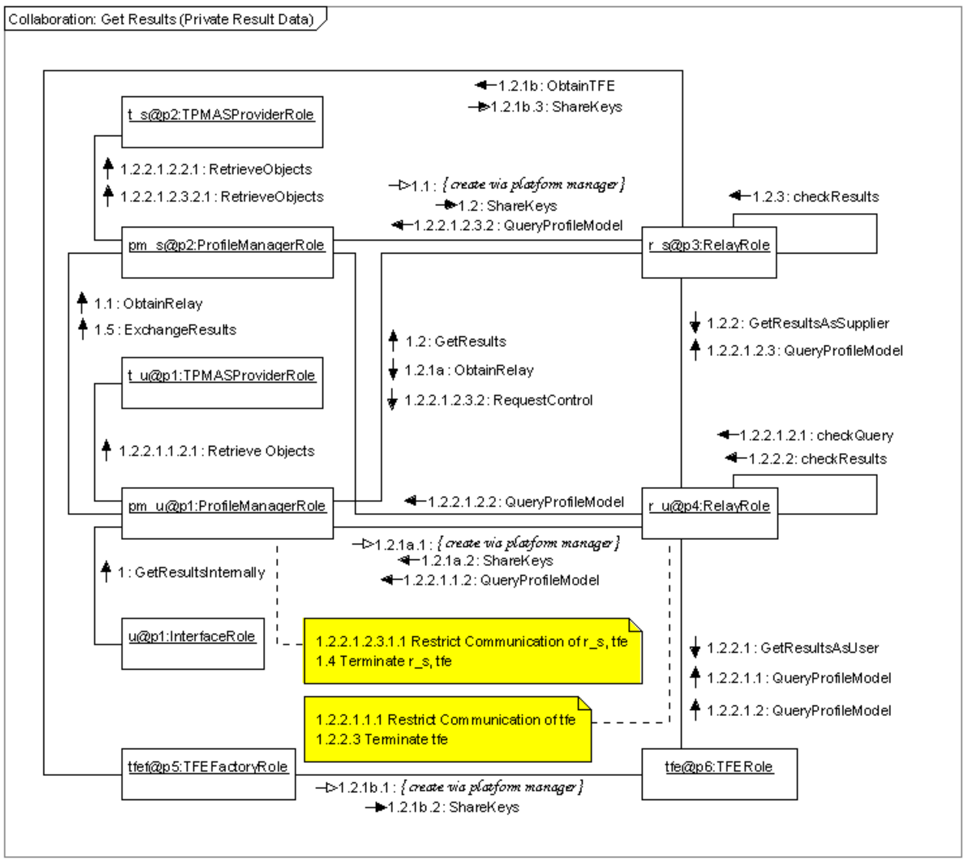

Collaboration: Get Results (Private Result Data)
1.2.1b : ObtainTFE
1.2.1b.3 : ShareKeys
t_s@p2:TPMASProviderRole
1.2.2.1.2.2.1 : RetrieveObjects
1.2.2.1.2.3.2.1 : RetrieveObjects
1.1 : {create via platform manager}
1.2 : ShareKeys
1.2.2.1.2.3.2 : QueryProfileModel
1.2.3 : checkResults
pm_s@p2:ProfileManagerRole
r_s@p3:RelayRole
1.1 : ObtainRelay
1.5 : ExchangeResults
1.2 : GetResults
1.2.1a : ObtainRelay
1.2.2.1.2.3.2 : RequestControl
1.2.2 : GetResultsAsSupplier
1.2.2.1.2.3 : QueryProfileModel
t_u@p1:TPMASProviderRole
1.2.2.1.1.2.1 : Retrieve Objects
1.2.2.1.2.1 : checkQuery
1.2.2.2 : checkResults
1.2.2.1.2.2 : QueryProfileModel
pm_u@p1:ProfileManagerRole
r_u@p4:RelayRole
1.2.1a.1 : {create via platform manager}
1.2.1a.2 : ShareKeys
1.2.2.1.1.2 : QueryProfileModel
1 : GetResultsInternally
u@p1:InterfaceRole
1.2.2.1.2.3.1.1 Restrict Communication of r_s, tfe
1.4 Terminate r_s, tfe
1.2.2.1 : GetResultsAsUser
1.2.2.1.1 : QueryProfileModel
1.2.2.1.2 : QueryProfileModel
1.2.2.1.1.1 Restrict Communication of tfe
1.2.2.3 Terminate tfe
tfef@p5:TFEFactoryRole
tfe@p6:TFERole
1.2.1b.1 : {create via platform manager}
1.2.1b.2 : ShareKeys

Figure 7.5: Collaboration Diagram for the main use cases "get recommendations" and "get prediction for item", based on private result data in a Recommender System context.

to its size. This may not apply to the mixed IR/IF scenario in which a constrained provider profile is used that is obtained via an additional non-privacy-critical query. Regular queries on the supplier profile (including the supplier profile models), however, are potentially critical with regard to user privacy, because the TFERole may use parts of the user profile within the query structure. This course of action should not be prevented completely, because it is actually the only feasible way to obtain a partial supplier profile containing the relevant parts of a supplier profile, short of retrieving the entire profile, as the relevant parts are expected to be those that have some relation to the user profile.

When querying the supplier profile, the respective filtering algorithms have to take user privacy into account and use either unlinkable user profile elements in the query, or no user profile elements as such at all. Exemplary filtering techniques for both approaches are given in Chapter 9. In the case

Table 7.4: The Phase *II* interaction steps of the information filtering stage for scenarios based on unlinkable queries and for the use cases based on linkable result data. The RELAYROLE^{User} has to remain anonymous in all interactions with the PROFILEMANAGERROLE^{Supplier}.

Step	$Sender \rightarrow Receiver$	Message	part of interaction
repeat *II.a* to *II.e* $\forall\ pr \in PR_u$:			
II.a	$TFE \rightarrow RR(U)$	$q(pr)$	QueryProfileModel
II.b	$RR(U) \rightarrow PMR(S)$	$q(pr)$	QueryProfileModel
II.c	$PMR(S) \rightarrow RR(U)$	$\{PP_{q(pr)}\}_{K_P}$	QueryProfileModel
II.f	$PMR(S)$ restricts communication of $RR(U), TFE$		
II.g	$PMR(S) \rightarrow RR(U)$	K_P	QueryProfileModel
II.h	$RR(U) \rightarrow TFE$	$PP_{q(PR_u)}$	QueryProfileModel

of honest and honest-but-curious participants we assume the TFEROLE to actually use a privacy-preserving approach for querying. Threats originating from a malicious TFEROLE are addressed in Section 7.3.1.2.

Unlinkable queries have to be realized through anonymized interaction: The RELAYROLE sends single queries to the PROFILEMANAGERROLE^{Supplier} (or the RELAYROLE^{Supplier} in case of private result data), and receives the respective results. Because agents on controlled platforms cannot communicate anonymously, these interactions have to be carried out before control of the RELAYROLE is established. In order to protect the provider data, it is send in encrypted form by the PROFILEMANAGERROLE^{Supplier}, who provides the key only after control has finally been established, i.e. after the final anonymous interaction. Taken together, these steps (as listed in Table 7.4 and Table 7.5) replace the steps of Phase *II* of the abstract protocol.

The unlinkability of single queries obviously depends on the number of parallel interactions of different agents realizing the RELAYROLE with one supplier: If only one single filtering process takes place in a given time period, unlinkability is not achieved. Unfortunately, it is difficult for the user to come up with a realistic estimation of this number. There are three approaches for increasing the degree of unlinkability in case a low number of parallel interactions is suspected:

- The time period may be increased by deliberately delaying the single interactions. While the probability of parallel interactions rises with increasing length of the time period, this approach also results in increasing response times, which may be critical in case the user actively waits for results.

Table 7.5: The Phase *II* interaction steps of the information filtering stage for scenarios based on unlinkable queries and for the use cases based on private result data. The RelayRole^{User} has to remain anonymous in all interactions with the ProfileManagerRole^{Supplier}.

Step	*Sender* $\rightarrow$ *Receiver*	Message	part of interaction
repeat *II.a* to *II.c* $\forall\ pr \in PR_u$:			
II.a	$TFE \rightarrow RR(U)$	$q(pr)$	QueryProfileModel
II.b	$RR(U) \rightarrow RR(S)$	$q(pr)$	QueryProfileModel
II.c	$RR(S) \rightarrow PMR(S)$	$q(pr)$	QueryProfileModel
II.d	$PMR(S) \rightarrow RR(S)$	$\{PP_{q(pr)}\}_{K_P}$	QueryProfileModel
II.e	$RR(S) \rightarrow RR(U)$	$\{PP_{q(pr)}\}_{K_P}$	QueryProfileModel
II.d	$RR(S)$ restricts communication of $RR(U), TFE$		
II.e	$PMR(S) \rightarrow RR(S)$	K_P	QueryProfileModel
II.f	$RR(S) \rightarrow RR(U)$	K_P	QueryProfileModel
II.g	$RR(U) \rightarrow TFE$	$PP_{q(PR_u)}$	QueryProfileModel

- Additional interactions based on dummy queries may be initiated by the RelayRole.
- Entire dummy interactions of the type GetResults may be initiated by the ProfileManagerRole^{User}. This approach may be problematic with regard to the overall performance.

In the following, we assume that the number of parallel interactions is sufficiently large, which is a realistic assumption for systems handling a large number of users.

Result Data Propagation In both main use cases, the results, i.e. recommendations, similar users, or a prediction, have to be propagated along the cascade of controllers. In the Recommender System scenario, the supplier may obtain the personalized information as well, mainly in order to improve the quality of its information, based on data about information in high demand. In the distributed Hybrid IF System scenario, result information is usually not propagated to the other participating entity because it is neither required nor would it generally be possible to realize unlinkability in this case, because a user generally participates in parallel interactions to a much smaller extent compared to a provider.

As described in Section 4.1, there are four different cases with regard to the propagation of result data, which are realized by adjusting the steps of Phase *III* of the information filtering protocol:

- *Completely Linkable Result Data*: In this scenario, no adjustments are required.
- *Semi-Linkable Result Data*: In this scenario, the ProfileManagerRole^{User} has to remain anonymous in all interactions with the ProfileManagerRole^{Supplier}. The interaction steps as such do not have to be adjusted.
- *Semi-Private Result Data*: In this scenario, the result data is propagated to the ProfileManagerRole^{User} via an additional relay, the RelayRole^{Supplier}. In single anonymous interactions, the ProfileManagerRole^{User} propagates the result data to the ProfileManagerRole^{Supplier}.
- *Completely Private Result Data*: In this scenario, the final interaction step between ProfileManagerRole^{User} and ProfileManagerRole^{Supplier} is omitted.

Other protocols are conceivable, especially for the scenarios based on private result data. By encrypting the result data, it would be possible to forgo the ProfileManagerRole^{Supplier}. It will turn out, however, that these alternatives are less suitable in the case of malicious participants, which is addressed in the following section. Therefore, they are not examined further at this point.

7.2.3 Summary

In the analysis phase, we have defined basic interactions addressing all threats originating from honest-but-curious participants in our approach by countermeasures as shown in Table 7.6. These basic interactions have to be refined further in order to address threats originating from malicious participants. We combine this refinement with other tasks of the design phase, which is discussed in the following section.

7.3 Design & Implementation

This section describes the agents and agent services of the Recommender Module. The interactions defined in the previous sections have to be refined further in order to address threats originating from malicious participants, as well as other aspects. It turns out that the interactions as such are sufficient, but the interaction steps have to be extended in many cases. The

Table 7.6: Threats in PPIF with honest-but-curious participants, and countermeasures in our approach.

permanent acquisition of	by user	by supplier	by filter
user profile data	n/a	does not acquire linkable data permanently	TFEROLE is controlled by user
supplier profile data	RELAYROLE^{User} is controlled by supplier	n/a	TFEROLE is controlled by supplier
result data	n/a	only acquired as specified	TFEROLE is controlled by user

following section describes extensions as countermeasures for various threats originating from malicious participants. Subsequent sections describe extensions addressing other aspects, and also the agents and agent services.

7.3.1 Threats and Countermeasures

Malicious participants may deviate from the specified protocols in any conceivable way either in order to propagate private information related to another participant, or in order to alter or disrupt the overall interaction for other reasons. Because there is no possible way for malicious participants on controlled platforms to communicate with external parties, we are able to restrict the discussion of threats originating from malicious participants to the interactions taking place along the cascade of controllers up to the initiator, i.e. to interactions between roles as defined in the Section 7.2.2 of this chapter as shown in Figure 4.1 and Figure 4.2 for the use cases based on linkable result data and private result data respectively.

As there is no critical interaction between roles aggregated by the same abstract entities, the opportunities for deviations are limited: Attempts to establish an additional channel for propagating private information are easily detected and stopped, basically because these attempts would be registered as obvious deviations from the protocol. Therefore, critical malicious attempts are limited to the following two main aspects, which are discussed in the following sections: Altering queries on the supplier profile, and altering result data. We discuss modifications to the protocols which address these threats in the following, starting with the description of various protocols as building blocks for secure message forwarding.

7.3.1.1 Secure Message Forwarding

We introduce two generic protocols for secure message forwarding as building blocks that are applied in the following sections in order to eliminate threats based on altered result data and the use of subliminal channels.

Consider a scenario involving three entities A, B, and C. A and C are not able to communicate directly, but both are able to communicate with B, whom they do not trust. A and C, however, have been able to exchange unlimited information, including a shared key K_{AC}, earlier. B has a secret key K_B. A intends to propagate the message m to C, which has to be done via B. B may know the content of the message, but should not be able to alter it. On the other hand, B will only forward the message if it can be certain that no additional information is propagated via any subliminal channel (such as a key used by A). The message m itself is not considered to contain any hidden information. This restriction is somewhat problematic because it is conceivable that A and C have previously agreed on some code to be used in the message. However, B may modify the original message prior to the actual protocol steps until he is convinced that it does not contain any hidden information.

This scenario is generally known as the prisoners' problem [102], because a real-life analogy consists of two prisoners intending to communicate, which has to be done via a warden who insists on being able to access all messages in unencrypted form and will only forward messages if they contain information he has approved. The prisoners, on the other hand, want to prevent the warden from modifying the messages in an undetectable manner. The prisoners' problem has been introduced to motivate the use of subliminal channels [102], whereas in our solution for secure message forwarding the goal is to prevent the participants from using subliminal channels.

The most straightforward protocol for secure message forwarding is a protocol based on digital signatures, as listed in Table 7.7. However, digital signature schemes have been shown to contain subliminal channels [103] and are therefore unsuitable in this context.

We provide the following solution for a secure message forwarding protocol (designated SMF1): As listed in Table 7.8, a keyed-Hash Message Authentication Code (HMAC) of the message m is propagated by A in addition to the message itself in order to prevent undetectable modifications of the message (Step a). The HMAC is encrypted by B with a secret key, and the encrypted HMAC is propagated, along with a hash of the key (Step c). The key K_{AC} is sent to B in order to allow B to verify that the HMAC is not used as a subliminal channel, i.e. that it is actually an encrypted hash of the message m (Step d). Finally, the message itself is propagated by B, along

Table 7.7: A protocol for secure message forwarding based on digital signatures. $s_K(x)$ indicates a message x signed via a private key K. The corresponding public key may be used to verify the signature.

Step	*Sender → Receiver*	Message
a	$A \rightarrow B$	$m, s_{K_A}(h(m))$
b	B checks m	
c	B checks $h(m)$ via public key $K_{A'}$	
d	$B \rightarrow C$	$m, s_{K_A}(h(m))$
e	C checks $h(m)$ via public key $K_{A'}$	

with the key K_B, which allows C to verify that the message has not been altered (Step f).

Propagating the hash of the key K_B prevents B from altering the message to m' after Step d, which would be undetectable otherwise because B could choose a key $K_{B'}$ so that $\{\{h(m)\}_{K_{AC}}\}_{K_B} = \{\{h(m')\}_{K_{AC}}\}_{K_{B'}}$. The key K_{AC} cannot be used as a subliminal channel: If A and C agree on a number of different keys K_{AC_n} before the first step of the protocol, and A uses a specific key K_{AC_i} in order to propagate additional information, C has to try out various keys until a valid hash is obtained. This is not possible because C has to propagate K_{AC} before he obtains the encrypted hash. Thus, C would have to guess the correct key.

Table 7.8: A protocol for secure message forwarding (SMF1) based on a HMAC.

Step	*Sender → Receiver*	Message
a	$A \rightarrow B$	$m, \{h(m)\}_{K_{AC}}$
b	B checks m	
c	$B \rightarrow C$	$h(K_B), \{\{h(m)\}_{K_{AC}}\}_{K_B}$
d	$C \rightarrow B$	K_{AC}
e	B decrypts and checks $h(m)$	
f	$B \rightarrow C$	m, K_B
g	C checks $h(K_B)$	
h	C decrypts and checks $h(m)$	

Obviously, the keys K_B and K_{AC} may only be used once, which is unfortunate because it adds to the complexity of the communication required for key exchange. This drawback has to be put up with in order to achieve secure message forwarding. The encryption scheme used for encrypting the hash obviously has to be secure against known-plaintext attacks, because otherwise B may be able to obtain K_{AC} after Step a and subsequently alter m in an undetectable way. Additionally, the encryption scheme must not be commutative, i.e. a scheme where $\{\{m\}_{K_A}\}_{K_B} = \{\{m\}_{K_B}\}_{K_A}$ cannot be used for this protocol: In this case, B could alter m to m' and choose a key $K_{B'}$ so that $\{m\}_{K_B} = \{m'\}_{K_{B'}}$ and, because of the commutativity, $\{\{m\}_{K_{AC}}\}_{K_B} = \{\{m'\}_{K_{AC}}\}_{K_{B'}}$. By propagating $K_{B'}$ instead of K_B, B would cause C to unknowingly decrypt m' instead of m.

We additionally provide a slightly modified protocol (SMF2), listed in Table 7.9, in which the message itself is encrypted, instead of using an HMAC. This solution has a slightly lower communication complexity, but a higher computational complexity (as shown in Table 7.10). It is somewhat less suitable for cascading secure message forwarding involving more than three participants, but more suitable for secure iterative message forwarding, as described in the following. This modified protocol does not suffer from the vulnerabilities mentioned above, i.e. a commutative encryption scheme may be used here as well as an encryption scheme vulnerable against known-plaintext attacks (although both options are generally not recommended).

Table 7.9: A protocol for secure message forwarding (SMF2) based on a symmetric encryption scheme.

Step	*Sender* → *Receiver*	Message
a	$A \rightarrow B$	$\{m\}_{K_{AC}}$
b	$B \rightarrow C$	$h(K_B), \{\{m\}_{K_{AC}}\}_{K_B}$
c	$C \rightarrow B$	K_{AC}
d	B decrypts and checks m	
e	$B \rightarrow C$	K_B
f	C decrypts m	
g	C checks $h(K_B)$	

Cascading Secure Message Forwarding A generalized case of the protocol for secure message forwarding is the following: A message is to be forwarded securely along a cascade of participants A_1, B_1, .., A_{n-1}, B_{n-1}, A_n, B_n (with B_n being optional), in which each A_i shares keys with and trusts the other A_j, but does not trust any B_k, and vice versa. Every participant along the cascade has to be able to verify the message m, but must not be able to modify it. This is achieved by repeating the protocol steps introduced above for every three consecutive participants. For SMF1, this is especially efficient because the final step of the iteration $i-1$ may be merged with the first step of the iteration i, resulting in a reduced communication complexity because m only has to be propagated once. For SMF2, no such

Table 7.10: A comparison of the protocols for secure message forwarding in terms of communication complexity and computational complexity. The size of hashes and keys is constant and therefore almost insignificant in relation to the size of messages. The same applies with regard to the computational complexity of encrypting and decrypting hashes vs. messages. $I(x)$ denotes information of size x.

	SMF1	SMF2
communication complexity		
# of $I(m)$	2	2
# of $I(h)$	3	1
# of $I(K)$	2	2
computational complexity		
# of encryptions of $I(m)$	–	2
# of decryptions of $I(m)$	–	4
# of encryptions of $I(h)$	2	–
# of decryptions of $I(h)$	4	–
# of hashing operations on I(m)	3	–
# of hashing operations on I(K)	2	2

optimization is possible. Additionally, when used in this way, the restrictions described above regarding vulnerability against known-plaintext attacks and commutativity apply to SMF2 as well.

Secure Iterative Message Forwarding Another generalized case of the protocol for secure message forwarding is the following: A intends to forward a number of messages to C via B, but B must not be able to withhold the propagation of message m_i and still obtain subsequent messages m_{i+x} at the same time. SMF1 cannot be used for this task without modifications, because A would not be able to decide whether to start another iteration of the protocol. SMF2, on the other hand, may be used for this task because if C does not receive the information required to obtain m_i, he may refuse to proceed with subsequent iterations, which prevents B from being able to decrypt m_{i+x}. Finally, the protocol for cascading secure message forwarding may be combined with the protocol for secure iterative message forwarding. The protocol listed in Table 7.11 constitutes an example.

7.3.1.2 Altering Queries

Based on the protocols for secure message forwarding, we are now able extend the interactions described above in order to counter malicious threats.

Queries on a profile are relayed through the RelayRole associated with the opposing participating entity. The RelayRole therefore has to decide whether a query is used as specified, i.e. to retrieve data in a privacy-preserving way, or whether the query is used instead to propagate private information. In the most straightforward case, a complete profile is returned and the respective query does not contain any specific information at all. If single profile elements are used within queries, unlinkable interactions may be carried out as described in Section 7.2.2.3. In all other cases, the decision becomes more complicated. Ultimately, if the user does not trust the filter completely, filtering techniques based on more advanced query structures should not be used because the possibility of using the queries as subliminal channels cannot be ruled out completely.

7.3.1.3 Altering Result Data in Recommender Systems

Depending on the use case, the result data consists of a set of recommendations or similar users, or a single prediction of the relevance of an item. In the Recommender System scenario, participants may attempt to alter this result data for various purposes. In order to prevent a successful execution of these attempts, the interaction steps are extended as explained below and as listed in Table 7.11, Table 7.12, and Table 7.13. Result data may be withheld maliciously by the TFERole or any RelayRole. While this cannot be prevented, it constitutes neither a serious nor a probable threat, because the respective main abstract entity would not benefit from this action and therefore the motivation for deviating from the protocol in this manner is considered to be low. Other threats are examined in detail in the following.

TFERole Alters Result Data The TFERole may attempt to propagate private user information via the result data, or otherwise alter the result data in a way that is unfavorable for the user entity. As the filter entity would not benefit directly from this action, it makes sense only if filter and supplier collude, or if the filter entity intends to cause suspicion of a possible collusion, a scenario that seems somewhat far-fetched.

The TFERole may also alter the result data in a way that results in the user obtaining incorrect data, while the supplier obtains correct data, e.g. by returning result data according to a previously defined code (such as a code in which a recommendation x actually stands for recommendation y, or a

Table 7.11: The extended protocol steps for the propagation of result data in the Recommender System scenario, for the case of completely linkable result data. For semi-linkable result data, the ProfileManagerRole^{User} has to remain anonymous in interactions with the ProfileManagerRole^{Supplier}. Apart from this modification, the same steps may be used.

Step	*Sender* → *Receiver*	Message	part of interaction
[*O.a* to *O.c*: Key sharing]			
O.a	$TFEF \rightarrow TFE$	KA	ShareKeys
O.b	$TFEF \rightarrow PMR(S)$	KA	ShareKeys
repeat *O.c* $\forall$ $res \in RES$:			
O.c	$PMR(U) \rightarrow RR(U)$	KD_{res}	ShareKeys
[Phase *I* and Phase *II* as above]			
[*III.a* to *III.i*: SMF1 with modified final step for RES]			
III.a	$TFE \rightarrow RR(U)$	$RES, \{H(RES)\}_{KA}$	GetResultsAsUser
III.b	$RR(U)$ analyzes RES		
III.c	$RR(U)$ creates secret key KB		
III.d	$RR(U) \rightarrow PMR(S)$	$h(KB), \{\{H(RES)\}_{KA}\}_{KB}$	GetResultsAsSupplier
III.e	$PMR(S) \rightarrow RR(U)$	KA	GetResultsAsSupplier
III.f	$RR(U)$ decrypts and checks $h(RES)$		
III.g	$RR(U) \rightarrow PMR(S)$	KB	GetResultsAsSupplier
III.h	$PMR(S)$ checks $h(KB)$		
III.i	$PMR(S)$ decrypts $H(RES)$ (cannot check it yet)		
[*III.j* to *III.r*: SMF2 $\forall res \in RES$]			
repeat *III.j* $\forall$ $res \in RES$:			
III.j	$RR(U) \rightarrow PMR(S)$	$\{res\}_{KD_{res}}$	GetResultsAsSupplier
III.k	$PMR(S)$ creates secret key KE_{res}		
repeat *III.l* $\forall$ $res \in RES$:			
III.l	$PMR(S) \rightarrow PMR(U)$	$h(KE_{res}), \{\{res\}_{KD_{res}}\}_{KE_{res}}$	GetResults
repeat *III.m* to *III.r* $\forall$ $res \in RES$:			
III.m	$PMR(U) \rightarrow PMR(S)$	KD_{res}	GetResults
III.n	$PMR(S)$ decrypts and analyzes res		
III.o	$PMR(S)$ checks $H(RES)$ from above w.r.t $h(res)$		
III.p	$PMR(S) \rightarrow PMR(U)$	KE_{res}, $complete\_data(res)$	GetResults
III.q	$PMR(U)$ decrypts res		
III.r	$PMR(U)$ checks $h(KE_{res})$		
III.s	$RR(U)$ terminates TFE		
III.t	$PMR(S)$ terminates $RR(U)$		

Table 7.12: The extended protocol steps for the propagation of result data in the Recommender System scenario, for the case of semi-private result data. The user must remain anonymous in Step *III.t* and Step *III.u*.

Step	*Sender* → *Receiver*	Message	part of interaction
[*O.a* to *O.d*: Key sharing]			
O.a/b	$TFEF \rightarrow TFE/RR(S)$	KA	ShareKeys
O.c	$PMR(S) \rightarrow RR(S)$	KC (session-independent)	ShareKeys
O.d	$PMR(U) \rightarrow RR(U)$	KD	ShareKeys
[Phase *I* and Phase *II* as above]			
III.a	$PMR(U)$ restricts communication of $RR(S), TFE$		
[*III.b* to *III.j*: SMF1 for RES]			
III.b	$TFE \rightarrow RR(U)$	$RES, \{h(RES)\}_{KA}$	GetResultsAsUser
III.c	$RR(U)$ analyzes RES		
III.d	$RR(U)$ creates secret key KB		
III.e	$RR(U) \rightarrow RR(S)$	$h(KB), \{\{h(RES)\}_{KA}\}_{KB}$	GetResultsAsSupplier
III.f	$RR(S) \rightarrow RR(U)$	KA	GetResultsAsSupplier
III.g	$RR(U)$ decrypts and checks $h(RES)$		
III.h	$RR(U) \rightarrow RR(S)$	RES, KB	GetResultsAsSupplier
III.i	$RR(S)$ analyzes RES		
III.j	$RR(S)$ checks $h(KB)$; decrypts and checks $h(RES)$		
[*III.k* to *III.q*: SMF1 with modified final step for RES]			
III.k	$RR(U) \rightarrow RR(S)$	$\{H(RES)\}_{KD}$	GetResultsAsSupplier
III.l	$RR(S)$ creates secret key KE		
III.m	$RR(S) \rightarrow PMR(U)$	$h(KE), \{\{H(RES)\}_{KD}\}_{KE}$	GetResults
III.n	$PMR(U) \rightarrow RR(S)$	KD	GetResults
III.o	$RR(S)$ decrypts and checks $h(RES)$		
III.p	$RR(S) \rightarrow PMR(U)$	KE	GetResults
III.q	$PMR(U)$ checks $h(KE)$		
repeat *III.r* to *III.w* $\forall\ res \in RES$:			
III.r	$RR(S) \rightarrow PMR(U)$	res	GetResults
III.s	$PMR(U)$ decrypts and checks $h(res)$		
III.t	$PMR(U) \rightarrow PMR(S)$	res	ExchangeResults
III.u	$PMR(S) \rightarrow PMR(U)$	$complete\_data(res), \{h(res)\}_{KC}$	ExchangeResults
III.v	$PMR(U) \rightarrow RR(S)$	$\{h(res)\}_{KC}$	GetResults
III.w	$RR(S)$ decrypts and checks $h(res)$		
III.x	$RR(U)$ terminates TFE		
III.y	$RR(S)$ terminates $RR(U)$		
III.z	$PMR(U)$ terminates $RR(S)$		

Table 7.13: The extended protocol steps for the propagation of result data in the Recommender System scenario, for the case of completely private result data.

Step	*Sender* → *Receiver*	Message	part of interaction
[$O.a$ to $O.c$: Key sharing]			
$O.a/b$	$TFEF \rightarrow TFE/RR(S)$	KA	ShareKeys
$O.c$	$PMR(U) \rightarrow RR(U)$	KD	ShareKeys
[Phase I and Phase II as above]			
$III.a$	$PMR(U)$ restricts communication of $RR(S), TFE$		
[$III.b$ to $III.j$: SMF1 for RES]			
$III.b$	$TFE \rightarrow RR(U)$	$RES, \{h(RES)\}_{KA}$	GetResultsAsUser
$III.c$	$RR(U)$ analyzes RES		
$III.d$	$RR(U)$ creates secret key KB		
$III.e$	$RR(U) \rightarrow RR(S)$	$h(KB), \{\{h(RES)\}_{KA}\}_{KB}$	GetResultsAsSupplier
$III.f$	$RR(S) \rightarrow RR(U)$	KA	GetResultsAsSupplier
$III.g$	$RR(U)$ decrypts and checks $h(RES)$		
$III.h$	$RR(U) \rightarrow RR(S)$	RES, KB	GetResultsAsSupplier
$III.i$	$RR(S)$ analyzes RES		
$III.j$	$RR(S)$ checks $h(KB)$, decrypts and checks $h(RES)$		
[$III.k$ to $III.r$: SMF1 for RES]			
$III.k$	$RR(U) \rightarrow RR(S)$	$\{h(RES)\}_{KD}$	GetResultsAsSupplier
$III.l$	$RR(S)$ creates secret key KE		
$III.m$	$RR(S) \rightarrow PMR(U)$	$h(KE), \{\{h(RES)\}_{KD}\}_{KE}$	GetResults
$III.n$	$PMR(U) \rightarrow RR(S)$	KD	GetResults
$III.o$	$RR(S)$ decrypts and checks $h(RES)$		
$III.p$	$RR(S) \rightarrow PMR(U)$	RES, KE	GetResults
$III.q$	$PMR(U)$ checks $h(KE)$		
$III.r$	$PMR(U)$ decrypts and checks $h(RES)$		
$III.s$	$RR(U)$ terminates TFE		
$III.t$	$RR(S)$ terminates $RR(U)$		
$III.u$	$PMR(U)$ terminates $RR(S)$		

prediction k for a prediction $k+m$). While there does not seem to be any way to prevent this, it affects the quality and consistency of the result data and therefore will probably be noticed. Additionally, it does not immediately threaten user privacy.

These threats do not apply to the case of completely private result data, because in that case the result data is acquired permanently by the user entity only. In the case of semi-private, semi-linkable or completely linkable result data, the following solution applies:

As the RelayRole^{User} receives all result data before it is acquired permanently by the supplier, it is able to check the result data for suspicious information. While some attempts may be detected immediately, e.g. if user profile elements are used as recommendations, more sophisticated attempts based on subliminal channels have to be detected as well. The possibility of subliminal channels within the result data used to propagate private information increases with the complexity of the result data. For example, a vector containing several floating point decimals may be used more easily for encoding information than single boolean values. Because recommendations are always a subset of the supplier profile data, it is sufficient to return a short identifier as a recommendation, based on which the user may subsequently obtain the complete element.

For the same reason, predictions should be taken from a limited range of possible values, instead of allowing arbitrary values. Taken together, these steps minimize the possibilities for subliminal channels. The interaction between the ProfileManagerRole^{Supplier} and the ProfileManagerRole^{User} is extended by a final step in which the complete recommendation is returned if necessary. To prevent the supplier from returning arbitrary information as the complete recommendation, the identifier should be meaningful in itself, i.e. a string expressing a recognizable movie title would be preferable to an apparently arbitrary number.

Furthermore, the TFERole may attempt to propagate private supplier information via the result data, or otherwise alter the result data in a way that is unfavorable to the supplier, in a similar way as described above. This only makes sense if filter and user collude, or if the filter intends to cause suspicion of a possible collusion. In this case, the following solution applies:

As either the RelayRole^{Supplier} or the ProfileManagerRole^{Supplier} receives all result data before it is acquired permanently by the user, it is able to check the result data for suspicious information, in a similar manner as described above. The use of subliminal channels by the TFERole (such as $h(RES)$ or KA) is prevented implicitly by using the protocol for secure message forwarding. Table 7.14 summarizes the threats and countermeasures discussed in this paragraph.

Table 7.14: Countermeasures against the TFEROLE as a malicious participant. The user entity has to be able to analyze the results before they are obtained by the supplier entity (permanently or temporarily), and the supplier entity has to be able to analyze the results before they are permanently obtained by the user entity. This is actually accomplished in all protocols as summarized here.

internal action	linkable result data (see Table 7.11)	semi-private result data (see Table 7.12)	compl. private result data (see Table 7.13)
user entity analyzes result data via	$RR(U)$ Step *III.b*	$RR(U)$ Step *III.c*	$RR(U)$ Step *III.c*
supplier entity obtains/ analyzes result data via	$PMR(S)$ Step *III.n*	$RR(S)$ Step *III.i*	$RR(S)$ Step *III.i*
user entity obtains result data via	$PMR(U)$ Step *III.q*	$PMR(U)$ Step *III.r*	$PMR(U)$ Step *III.p*

Other Roles Alter Result Data The RELAYROLE^{User} may attempt to propagate private supplier information via the result data, or otherwise alter the result data in a way that is unfavorable to the supplier. In order to prevent a successful execution of this attempt, the result data is propagated via the protocol for secure message forwarding SMF1 (Step *III.a* to Step *III.i* in Table7.11, Step *III.b* to Step *III.j* in Table 7.12, and Step *III.b* to Step *III.j* in Table 7.13).

If the TFEROLE and the RELAYROLE^{User} collude, this threat obviously is not preventable at this stage. In this case, the supplier still is able to react in the same manner as described above for the threat originating from the TFEROLE. In all other cases, however, it is more efficient to rely on the encryption than having to analyze the returned data.

The RELAYROLE^{Supplier} or the PROFILEMANAGERROLE^{Supplier} may attempt to propagate private user information via the result data, or otherwise alter the result data in a way that is unfavorable to the user. In order to prevent a successful execution of this attempt, the result data is propagated via the protocol for secure message forwarding SMF2 (Step *III.j* to Step *III.r* in Table 7.11, and the protocol for secure message forwarding SMF1 respectively (Step *III.k* to Step *III.q* in Table 7.12, and Step *III.k* to Step *III.r* in Table 7.13). In particular, the iterative disclosure of result data in Step *III.j* to Step *III.r* in Table 7.11 prevents the PROFILEMANAGERROLE^{Supplier} from withholding result data. Similarly, the protocol part consisting of Step *III.r* to Step *III.w* in Table 7.12 prevents the PROFILEMANAGERROLE^{User} from withholding or altering result data, because this would be noticed by the RelayRole^{Supplier}, which subsequently would be able to halt the protocol.

7.3.1.4 Altering Result Data in Hybrid IF Systems

In the Hybrid IF System scenario, participants may attempt to alter result data for various purposes largely analogous to the cases described above. In order to prevent a successful execution of these attempts, the interaction steps are extended as listed in Table 7.15.

7.3.1.5 Completion of Iterative Disclosure

A minor but potentially problematic threat arises in protocols based on an iterative disclosure of result data: While a participant withholding the entire result data would cause the respective protocol to stop, withholding the final part of the result data (e.g. the final recommendation) does not have any direct consequences because the sender does not subsequently receive any additional information anyway.

If both participants act strictly rationally, this problem does not only affect the final part of the result data, but ultimately the entire result data, because a participant who cannot expect to receive data in step n may choose not to carry out step $n-1$, and thus no participant would be sufficiently motivated even to begin the protocol.

We discuss this threat for the different scenarios and cases:

- In the cases of completely linkable data and semi-linkable data in the Recommender System scenario, this threat is less problematic because it only applies to the ProfileManagerRole^{Supplier} (Step *III.p* in Table 7.11), who can be expected to carry out the protocol as specified in order to gain the trust of the users.

- In the case of semi-private result data, this threat is less problematic because while the ProfileManagerRole^{User} may actually withhold the final result data (Step *III.t* in Table 7.12), this does not lead to the ProfileManagerRole^{Supplier} withholding the previous data, because the interactions are unlinkable from the point of view of the ProfileManagerRole^{Supplier}.

- In the case of completely private result data in the Recommender System scenario, the threat does not apply because there is no iterative disclosure.

Table 7.15: The extended protocol steps for the propagation of result data in the Hybrid System scenario, for the case of completely private result data.

Step	*Sender* $\rightarrow$ *Receiver*	Message	part of interaction
[$O.a$ to $O.e$: Key sharing]			
$O.a/b$	$TFEF \rightarrow TFE/RR(S)$	KA	ShareKeys
$O.c$	$PMR(S) \rightarrow RR(S)$	KC	ShareKeys
repeat $O.c$ $\forall$ $res(U) \in RES(U)$:			
$O.d$	$PMR(S) \rightarrow RR(S)$	$KF_{res(U)}$	ShareKeys
$O.e$	$PMR(U) \rightarrow RR(U)$	KD	ShareKeys
[Phase I and Phase II as above]			
$III.a$	$PMR(U)$ restricts communication of $RR(S), TFE$		
$III.b$	$RR(S) \rightarrow RR(U)$	KC	GetResultsAsSupplier
repeat $III.c$ $\forall$ $res(U) \in RES(U)$:			
$III.c$	$RR(S) \rightarrow RR(U)$	$KF_{res(U)}$	GetResultsAsSupplier
[$III.d$ to $III.k$: SMF1 for RES]			
$III.d$	$TFE \rightarrow RR(U)$	$RES, \{h(RES)\}_{KA}$	GetResultsAsUser
$III.e$	$RR(U)$ analyzes RES; creates secret key KB		
$III.f$	$RR(U) \rightarrow RR(S)$	$h(KB), \{\{h(RES)\}_{KA}\}_{KB}$	GetResultsAsSupplier
$III.g$	$RR(S) \rightarrow RR(U)$	KA	GetResultsAsSupplier
$III.h$	$RR(U)$ decrypts and checks $h(RES)$		
$III.i$	$RR(U) \rightarrow RR(S)$	RES, KB	GetResultsAsSupplier
$III.j$	$RR(S)$ analyzes RES		
$III.k$	$RR(S)$ checks $h(KB)$; decrypts and checks $h(RES)$		
[$III.l$ to $III.r$: SMF1 for $RES' = \bigcup res'$ with $res' \overset{def}{=} \{res(S)\}_{KC}, \{res(U)\}_{KF_{res(U)}}$]			
$III.l$	$RR(U) \rightarrow RR(S)$	$\{h(RES')\}_{KD}$	GetResultsAsSupplier
$III.m$	$RR(S)$ creates secret key KE		
$III.n$	$RR(S) \rightarrow PMR(U)$	$h(KE), \{\{h(RES')\}_{KD}\}_{KE}$	GetResults
$III.o$	$PMR(U) \rightarrow RR(S)$	KD	GetResults
$III.p$	$RR(S)$ decrypts and checks $h(RES')$		
$III.q$	$RR(S) \rightarrow PMR(U)$	RES', KE	GetResults
$III.r$	$PMR(U)$ checks $h(KE)$; decrypts and checks $h(RES')$		
repeat $III.s$ to $III.t$ $\forall$ $res' \in RES'$:			
$III.s$	$PMR(U) \rightarrow PMR(S)$	$\{res(S)\}_{KC}$	ExchangeResults
$III.t$	$PMR(S) \rightarrow PMR(U)$	$KF_{res(U)}$	ExchangeResults
$III.u$	$RR(U)$ terminates TFE		
$III.v$	$RR(S)$ terminates $RR(U)$		
$III.w$	$PMR(U)$ terminates $RR(S)$		

- In the case of completely private result data in the Hybrid System scenario, the threat may be countered by letting the initiating user represented by the ProfileManagerRole^{User} decide on the size of the result data (i.e. the number n of recommendations to be returned), and by keeping this number secret from the other user represented by the ProfileManagerRole^{Supplier}. Thus, the ProfileManager-Role^{Supplier} cannot withhold (Step *III.t* in Table 7.15) without risking to miss information about further result data.

If the result data consists of a single prediction, the iterative disclosure procedure is reduced to a single iteration which is even more problematic. Therefore, if predictions are to be returned instead of recommendations, the result data should contain a number of predictions for different items instead of a single prediction. If this is done, the countermeasures described above apply here as well.

7.3.1.6 Reaction to Detected Threats

Finally, all roles involved in the interactions have to be able to react in an appropriate manner if they detect deviations from the protocol. While the ProfileManagerRole may just abort the overall interaction and log or otherwise report the attempted deviation, the RelayRole is more restricted in this regard, because it cannot communicate freely. As an example, if the RelayRole in the Recommender System scenario detects or suspects a collusion between the other roles involved, it cannot report to the ProfileManagerRole^{User}. It may still abort the overall interaction, but in this case the supplier may divert suspicion by announcing that a different problem caused the interruption of the overall interaction.

Therefore, a RelayRole should be allowed to establish an additional channel in order to propagate some kind of status flag to the ProfileManagerRole^{User}. If only a few bits are used for this channel, it is likely to be too small to be used as a feasible subliminal channel, but wide enough to communicate one of a number of status flags previously agreed upon. If the meaning of status flags is changed in each overall interaction, the supplier cannot modify the status flag in an undetectable way. Additionally, the RelayRole should continue to interact with the supplier even when it detects a deviation in order to ensure that the result flag is actually propagated, but at the same time it should deviate itself from the protocol in order to prevent the acquision of private information, e.g. by replacing encrypted result data with random noise.

7.3.2 Other Requirements

Apart from privacy requirements, interactions may be extended for other reasons, e.g. in order to improve the performance of the system:

In the mixed IR/IF scenario, recommendations may be generated in two ways:

- The user may apply the IR-related query to recommendations retrieved in the usual manner. However, this approach is likely to result in an small or even empty set of recommendations, because few or no matches may be found in the candidate set of recommendations, which is usually itself rather small.

- The supplier may apply the IR-related query to his profile, and generate recommendations from the result set. While this approach cannot utilize a pre-computed profile model, it is actually more feasible because the relevant part of the provider profile can be expected to be small enough to be propagated completely to the TFRRole.

Therefore, the interaction GetRecommendations has to be extended by using the IR-related query as an additional input parameter.

Furthermore, especially in this case repeated interactions between a given user and supplier within a short time period are likely because the user may send various IR-related queries[5]. In this case, it is not required to repeat all interactions for each single filtering process, because the agents on controlled platforms may be re-used. Interactions may be extended accordingly to allow this re-use of agents. We omit the details of the extended interactions because they are not directly relevant for the overall architecture. It should be noted, however, that re-using the RelayRole rules out the possibility of anonymous interactions for querying the provider profile.

7.3.3 Agents and Agent Services

Based on the extended interactions described in the previous sections, we are now able to define the agent services. In most cases, interactions are mapped to agent services in a straightforward manner, as shown in Table 7.16. Interactions with similar input and output are aggregated as one agent service, because they can be regarded as having the same effect. Depending on the actual interaction, different protocols are used within the respective service.

[5]In the regular scenario, repeated interactions only make sense when at least one profile changes, which happens only intermittently.

Table 7.16: The mapping of interactions to agent services.

Interaction	Table	Agent Service
Information Collection Stage		
UpdateProfile	A.21	*UpdateProfile*
QueryProfile	A.22	*QueryProfile*
Information Processing Stage		
ObtainTFE	A.23	*ObtainTFE*
SetUpdatePolicy	A.24	*SetUpdatePolicy*
UpdateProfileModel	A.25	*UpdateProfileModel*
QueryProfileModel	A.26	*QueryProfileModel*
ModifyProfileModel	A.27	*ModifyProfileModel*
Information Filtering Stage		
GetResultsInternally GetResults GetResultsAsSupplier GetResultsAsUser	A.28 A.29 A.30 A.31	*GetResults*
ExchangeResults	A.32	*ExchangeResults*
ObtainRelay	A.33	*ObtainRelay*
ShareKeys	A.34	*ShareKeys*

Roles are aggregated by agents in a similarly straightforward manner, as shown in Table 7.17. While it may be advisable to aggregate various roles belonging to the same abstract entity (such as the TPMASProviderRole^{User} and the ProfileManagerRole^{User}) for performance reasons, we use separate agents in order to keep the architecture flexible.

Table 7.17: The mapping of roles to agents.

Role	Agent
InterfaceRole^{User}	**InterfaceAgentUser**
InterfaceRole^{Provider}	**InterfaceAgentProvider**
ProfileManagerRole^{User}	**PMAgentUser**
ProfileManagerRole^{Provider}	**PMAgentProvider**
RelayRole^{User}	**RelayAgentUser**
RelayRole^{Provider}	**RelayAgentProvider**
TFEFactoryRole	**TFEFactoryAgent**
TFERole	**TFEAgent**

Communication within agent services is encrypted by mechanisms provided by the respective MAS architecture, unless the service is considered not to require encryption, either because no sensitive information is communicated, or because the respective data is already encrypted for other reasons, as described above.

7.3.4 Implementation

As we have only specified agents and agent services for this module, the implementation is straightforward and therefore most details are omitted here. Advanced Encryption Standard (AES) is used as the symmetric encryption scheme and HMAC-SHA-1 as the MAC based on a cryptographic hash function. These algorithms may easily be replaced with similarly suited algorithms.

For load balancing and improved performance, the provider entity may use a different agents providing the same functionality, i.e. realizing the same role, such as a number of **PMAgentProvider** agents. In this case, the load balancing must be actually carried out internally, e.g. by using a single **PMAgentProvider** manager agent distributing the actual load. It must not be realized by providing a dedicated **PMAgentProvider** agent whenever a user initiates a filtering process because in this case an honest-but-curious provider may link anonymous communications with the **PMAgentProvider** as recipient to a specific user. In the basic implementation, load balancing is not addressed.

7.4 Summary

This chapter describes functionality provided by the Recommender Module addressing the use cases "get prediction for item" and "get recommendations", as well as the use cases "update profile elements" and "update profile model", as defined in Section 4.1. In addition to the use of the Recommender Module functionality in a Recommender System context, it also covers its use in a Hybrid IF System context.

We briefly motivate the need for the Recommender Module (Section 7.1). We specify an ontology containing the basic concepts (Section 7.2.1). We specify roles and basic interactions for the three stages information collection, information processing, and information filtering, and address threats arising from honest-but-curious participants, in particular in the context of query data propagation and result data propagation (Section 7.2.2). Regarding the design and implementation of the specified functionality, we address threats arising from malicious participants by specifying two basic protocols for secure message forwarding (Section 7.3.1.1). Based on these protocols, we address the threats of altering queries (Section 7.3.1.2), altering result data

(Section 7.3.1.3 and (Section 7.3.1.4), and threats in the context of iterated disclosure of results (Section 7.3.1.5). We briefly discuss how roles should react when a threat is detected (Section 7.3.1.6), and how interactions have to be extended in order to meet other requirements (Section 7.3.2). Finally, we list agents and agent services (Section 7.3.3), and we discuss implementation details (Section 7.3.4). The following chapter addresses the remaining main use cases of our approach.

Chapter 8

The Matchmaker Module

This chapter describes functionality provided by the Matchmaker Module, i.e. it primarily addresses the use cases "get prediction for user" and "get similar users", as defined in Section 4.1. It uses functionality of the Recommender Module described in the previous chapter.

The chapter is structured as follows: Section 8.1 briefly motivates the Matchmaker Module. Section 8.2 describes the ontologies, roles and interactions of the module, while Section 8.3 describes the agents and agent services realizing these interactions. Section 8.4 concludes the chapter with a summary.

8.1 Motivation

The Matchmaker Module is one of the two core modules of our approach for Privacy-Preserving Information Filtering. Together with the Recommender Module, it addresses all use cases defined in Section 4.1. It provides primary functionality related to the requirement of user privacy (see Table 4.2). Thus, the need for functionality described in this chapter is motivated directly by the outline of our solution given in Section 4.2, as the abstract IF protocols introduced in the outline are realized via agent interactions, i.e. as part of agent services.

8.2 Analysis

This section describes the ontologies, roles and interactions of the Matchmaker Module. For the sake of readability, all tables and diagrams specifying these components may be found in Appendix A.4.

8.2.1 Ontologies

The main ontology of this module, the ontology "Distributed Information Filtering" shown in Figure A.6 of the Appendix, complements the ontology introduced in the previous chapter. It contains categories and attributes for distributed Matchmaker Systems and distributed Hybrid IF Systems that are explained further in the context of the interactions they are used in.

8.2.2 Roles and Interactions

The Matchmaker Module utilizes the roles introduced in Table 7.1, and one additional role as described in Table 8.1. For the role schema, see Appendix A.4.

Table 8.1: The roles participating in the Matchmaker Module.

role name	short name/ aggregated by user provider filter
InterfaceRole ProfileManagerRole RelayRole TPMASProviderRole TFERole TFEFactoryRole	see Table 7.1
CentralizedModelManagerRole	CMMR(P)
Responsible for the management of relations between profile elements and references to user entities.	

Analogous to the Recommender Module, we initially assume all participating roles to act in an honest or at least honest-but-curious manner, and address threats emanating from roles acting in a malicious manner in Section 8.3.1.

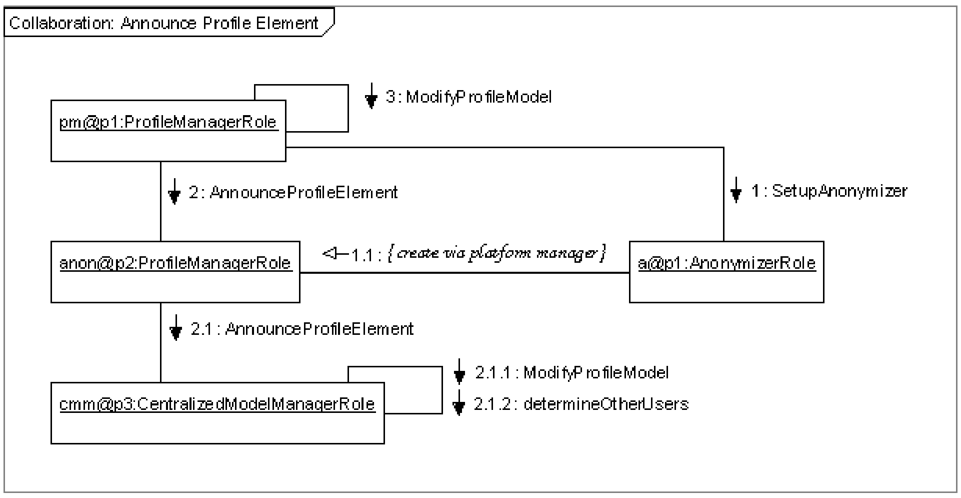
8.2.2.1 Determining Potentially Similar Users

In distributed collaboration-based IF approaches, similar users are determined by comparing user profiles. In our approach, the user profiles are distributed among the user entities. Therefore, similar users have to be determined in a distributed manner as well. Assuming the Information Filtering process of determining the similarity of two specific users as given, the straightforward approach of determining all similar users would be to apply this Information Filtering process to all combinations of two users. In systems with a large number of users, however, it is obviously infeasible to carry out the Information Filtering process for every pair of users, because the overall complexity would be quadratical in the number of users not even taking into account the fact that the process has to be repeated periodically because profile elements and thus similarities change over time. Therefore, the Information Filtering process should only be carried out for pairs of users who can be expected to be similar with a probability that is at least above average. This section introduces interactions for determining such candidate pairs. Other approaches are discussed in Section 3.2.4.

Candidate pairs are determined during the information collection and information processing stage. They are stored in the user profile models of the respective candidate users. We determine candidate pairs by considering two users as potentially similar if they have profiles containing the same profile element, or similar profile elements.

An entity keeping track of the profile elements of different user profiles would be able to determine overlaps and thus potentially similar users. In our approach, however, all user profile information is stored in a decentralized way by the user agents. Therefore, an additional role realizing this task is introduced, namely the CentralizedModelManagerRole. While this role could be associated with any abstract entity, the provider entity is the most obvious entity for aggregating this role, mainly because it already manages the items that are potential user profile elements. Additionally, the provider entity may be sufficiently motivated to carry out the tasks assigned to this role because the respective interactions allow the provider entity to collect general information about the dissemination of the information it provides, such as statistics of the most popular items, i.e. items appearing in a large number of user profiles.

In relation to a given profile element, the CentralizedModelManagerRole stores references to users who have added this element to their respective profiles within a profile model as part of the information processing stage. As a user entity may add an element to its profile without interacting with any other role, the user entity itself is responsible for announcing the operation to the CentralizedModelManagerRole, as an additional interaction within the main use case "update profile model". The CentralizedModelManagerRole utilizes a filtering technique in order to determine references to other user entities who have announced this element or similar elements, and notifies the user entity as an additional interaction within the main use case "update profile model". The other user entities do not have to be notified by the CentralizedModelManagerRole, because they will be contacted by the user entity itself if it actually intends to determine similar users. The respective interaction AnnounceProfileElement



Collaboration: Announce Profile Element
pm@p1:ProfileManagerRole
3 : ModifyProfileModel
2 : AnnounceProfileElement
1 : SetupAnonymizer
anon@p2:ProfileManagerRole
1.1 : {create via platform manager}
a@p1:AnonymizerRole
2.1 : AnnounceProfileElement
cmm@p3:CentralizedModelManagerRole
2.1.1 : ModifyProfileModel
2.1.2: determineOtherUsers

Figure 8.1: Collaboration Diagram for the partial use case "announce profile element" as part of the main use case "update profile model".

is specified in Table A.40 of the appendix.

The obvious problem arising from the introduction of the CentralizedModelManagerRole is the privacy of the user entities, i.e. the fact that user profile data is stored in a centralized way by a potentially honest-but-curious role. This problem is addressed by the following solution: The reference to the user entity stored by the CentralizedModelManagerRole does not contain data that may be used to actually identify the respective user entity. Instead, a pseudonym is stored which may be used to contact the respective entity. Furthermore, different pseudonyms have to be used by a given user entity for different profile elements, because otherwise the complete user profile could be reconstructed by searching for all profile elements associated with the same pseudonym. The mechanism for anonymous communication introduced in Section 5.2 is used to contact the user entity via its pseudonym. In other words, the user entity has to utilize an AnonymizerRole in order to achieve receiver anonymity when contacted by potentially similar user entities, and for achieving sender anonymity when announcing the respective profile element. The interaction between the different roles is shown in Figure 8.1.

Due to the fact that all data stored in the centralized model is anonymized, all threats originating from an honest-but-curious provider entity may be addressed adequately, because there is no way for the provider entity to obtain additional private information. Threats originating from malicious participants are addressed in the design phase described in Section 8.3.

8.2.2.2 Determining the Actual Similarity

Determining the actual similarity of two users is a rather straightforward task. In fact, we have already described all required interactions in the previous chapter, because the use case “get prediction for user” is basically realized via the same interaction protocol as the use cases “get prediction for item” and “get recommendations”, with the following differences:

- The supplier entity is actually realized by a second user entity, instead of a provider entity.
- User profiles are usually small enough to be processed entirely. Therefore, the RelayRole does not have to actually analyze queries (as long as it ascertains that no private data is used within the query structure).
- The filtering technique is primarily applied in order to determine the overall similarity of the two user profiles. It may additionally provide recommendations and/or predictions within the same interaction. Obviously, recommendations are generally only likely to be relevant in cases of high similarity. Filtering techniques may be designed in a way that allows their use for both goals. We give an example of a suitable filtering technique in the following chapter.

The use case “get similar users” is realized by carrying out the interactions of the use case “get prediction for user” for each candidate user, whereas the top-N most similar users are retained.

As we utilize no additional interactions for determining the actual similarity of users, no additional threats originating from honest-but-curious participants in this contexts have to be addressed.

8.2.2.3 Hybrid IF System Functionality

A Hybrid IF System generates recommendations and predictions of items via determining similar users. It may be realized simply by combining Matchmaker System functionality and Recommender System functionality. In our approach, the functionality described above allows a user to find other similar users. Recommendations or predictions of item relevance may be obtained from a similar user by carrying out the interactions described in the previous chapter, with the similar user representing the supplier. Both steps may also be combined by returning recommendations or predictions of item relevance in addition to the value indicating the similarity of users, which reduces the number of interactions between users. In this case, the additional result data should only be returned if the users are actually similar, because otherwise the result data cannot be expected to be relevant. It should also be noted that while result data from two user profiles could theoretically be generated via the same filtering techniques that are used in the pure Recommender System context, using adjusted filtering techniques in this context may be more advantageous with regard to quality.

8.3 Design & Implementation

This section describes the agents and agent services of the Matchmaker Module. The interactions defined in the previous sections have to be refined further in order to address threats originating from malicious participants, as well as further requirements. The following section describes extensions constituting countermeasures for various threats originating from malicious participants. Subsequent sections describe extensions addressing further requirements, and finally the agents and agent services.

8.3.1 Threats and Countermeasures

The introduction of the CentralizedModelManagerRole creates new possibilities for critical malicious actions. A malicious entity acting as a user entity may attempt to reconstruct the user profile of other user entities by the following course of action: The malicious entity adds a large number of elements to its user profile, and thus receives a subsequently large number of references to potentially similar user entities. By determining the similarities of the respective user entities (which are initially only known via their pseudonyms), the malicious entity may subsequently be able to link different pseudonyms to one single user entity, namely in cases where the similarity value and/or the generated recommendations and predictions are exactly identical. Once different pseudonyms are linkable to a single user entity, the respective profile elements become linkable as well, and thus a user profile may be reconstructed.

This threat is basically prevented by blurring the returned similarity value, making it impossible to link similarities[1]. Recommendations should

[1]As an example, if similarity is expressed as a continuous value between 0% and 100%, two separate similarities of 89.1327% probably indicate the same actual user, while using discrete values such as 0%, 5%, 10%, etc. would not allow to draw the same conclusion for two similarities of 90%, because the value will apply to many different users.

only be returned in cases of high similarity and non-suspicious foreign user profiles. User profiles containing only a few elements, or an exceedingly large number of elements, should be considered suspicious as they are likely not to belong to a regular, i.e. honest user entity.

Malicious actions with regard to the propagation of the similarity value itself do not have to be addressed explicitly, because it may be treated as a special kind of recommendation in this respect, and the mechanisms described in Section 7.3.1.3 apply, with the following exception: There is actually no way to prevent a ProfileManagerRole from withholding the similarity value itself. Because the similarity value alone does not give any useful information, this threat is negligible.

8.3.2 Other Requirements

Apart from privacy requirements, interactions may be extended for other reasons, e.g. in order to improve the performance of the system:

As described in Section 5.2, using relay agents for achieving receiver anonymity is problematic because these relay agents are not as short-lived as relay agents used for achieving sender anonymity, basically because the receiver usually does not decide when the communication takes place. Therefore, in a straightforward realization of the interaction introduced above, each user entity has to create and maintain one additional agent per user profile element. However, the total number of agents would be rather large in systems with a large number of users and comprehensive user profiles ($O(s \cdot u)$ for u users with an average number of elements s). This number could be reduced to $O(u)$, which is manageable because there are already $O(u)$ agents associated with user entities anyway, by the following approach:

Based on the average user profile size (which may be determined via the CentralizedModelManagerRole), a correspondingly large number of time slots are introduced, and each potential user profile element is assigned a time slot. User entities announce user profile elements only during the respective time slot, and thus have to keep the respective agents realizing the functionality for receiver anonymity alive only during that time slot.

If the suggested approach is still considered infeasible for a specific implementation, other solutions may be used without having to change the overall architecture: Solutions related to mix networks or similar anonymizer approaches (for which see Section 3.1.1) may be successfully applied in this case, although it should be noted that they would introduce additional trust issues.

8.3.3 Agents and Agent Services

Based on the extended interactions described in the previous sections, we are now able to define an additional agent service. As in most other cases, the respective interaction is mapped to the agent service in a straightforward manner, as shown in Table 8.2. Roles are aggregated by agents in a similarly straightforward manner, as shown in Table 8.3.

Table 8.2: The mapping of interactions to agent services.

Interaction	Table	Agent Service
AnnounceElement	A.40	*AnnounceElement*

Table 8.3: The mapping of roles to agents.

Role	Agent
CentralizedModel-ManagerRole	**CentralizedModel-ManagerAgent**

8.3.4 Implementation

As we have only specified agents and agent services for this module, the implementation is straightforward and therefore its details are omitted here.

8.4 Summary

This chapter describes functionality provided by the Matchmaker Module addressing the use cases "get prediction for user" and "get similar users", as defined in Section 4.1. In addition to the use of the Matchmaker Module functionality in a distributed Matchmaker System context, it also covers its use in a distributed Hybrid IF System context.

We briefly motivate the need for the Matchmaker Module (Section 8.1). We specify an ontology containing the basic concepts (Section 8.2.1). We specify roles and basic interactions, and address threats arising from honest-but-curious participants 8.2.2). Regarding the design and implementation of the specified functionality, we address threats arising from malicious participants (Section 8.3.1), and we discuss how interactions have to be extended in order to meet other requirements (Section 8.3.2). Finally, we list agents and agent services (Section 8.3.3). The following chapter describes exemplary filtering techniques that may be used in our approach.

Chapter 9

Exemplary Filtering Techniques

This chapter describes exemplary filtering techniques that may be used by the Recommender Module and the Matchmaker Module. These filtering techniques are provided as building blocks to be utilized by the filter entity. While the modules described in the previous chapters provided ontologies, interactions and ultimately agents and agent services, this chapter deals with functionality to be used internally, i.e. within a single agent.

As the Gaia methodology is not applicable to internal functionality, the section of this chapter are structured differently than the sections of the previous chapters. Section 9.1 briefly motivates exemplary filtering techniques. Section 9.2 describes the general requirements of filtering techniques that are to be applied in our approach for Privacy-Preserving Information Filtering, and Section 9.3 describes three exemplary filtering techniques meeting these requirements. Section 9.4 concludes the chapter with a summary.

9.1 Motivation

In the previous chapters, the filtering techniques used by the Recommender Module and the Matchmaker Module have been largely treated as black boxes, i.e. we have assumed that filtering techniques exist which are may be utilized within the information processing stage and the information filtering stage in a way that meets all functional and non-functional requirements.

In this chapter, we break down the requirements regarding the use of a specific filtering technique, and we describe exemplary filtering techniques in order to show that these requirements may actually be met.

9.2 Analysis

This section lists the requirements for filtering techniques to be used in the context of Privacy-Preserving Information Filtering. It should be noted that our approach does not require a single filtering technique to be applicable in the context of each of the four main use cases, or each of the sub-cases for the propagation of result data (as defined in Section 4.1). However, all use cases should be covered by at least one filtering technique.

As discussed in Section 2.2.1, there are two main groups of filtering techniques, namely feature-based and collaboration-based filtering techniques.

Feature-based filtering techniques are less problematic with regard to privacy because the respective profiles do not contain private data associated with other users. There are, however, feature-based approaches that are not directly applicable: Learning-based approaches in which the filter entity uses the data obtained during a specific filtering process (or data provided by the user as feedback) in order to refine further filter processes are obviously problematic because our approach does not allow any additional data to be propagated by the Temporary Filter Entity. Therefore, if these approaches are to be used the feedback has to be obtained outside of the PPIF part of the respective system, e.g. by using the same filtering technique in a non-privacy-preserving context.

Collaboration-based filtering techniques are generally problematic because they are based on data obtained by combining and analyzing elements from different user profiles. They may still be used either by again obtaining the data outside of the PPIF part of the respective system, or by combining a feature-based and a collaboration-based approach, e.g. as described as our solution for Matchmaker Systems in Chapter 8. In the latter case, however, the actual algorithm used to create the centralized model during the information processing stage, as well as the actual algorithm used to generate result data based on two profiles during the information filtering stage, are feature-based filtering techniques.

There are two main aspects that have to be considered for filtering techniques to be applied in our approach:

- *Influence of supplier profile data on obtained user profile data*: The protocols described in Chapter 7 are only applicable if the utilized filtering algorithm does not have to retrieve user profile data based on obtained supplier profile data, because as soon as supplier profile data has been obtained, no further user profile data may be obtained in a privacy-preserving manner. While the protocols could be extended to facilitate an iterative propagation of user and supplier profile data, this is usually unnecessary because the user profile is expected to be small enough to be propagated as a whole.

- *Influence of user profile data on obtained supplier profile data*: In practice, the applicability of a specific filtering technique largely depends on the size of the provider profile, including the models maintained during the information processing stage: While the user profile is typically small enough to be propagated entirely, there are different options for the propagation of the provider profile, in case the supplier is actually a provider entity (and not a different user entity):
  - *Complete Propagation*: Propagating the entire provider profile is usually infeasible, due to its size. The situation is analogous to the Private Information Retrieval scenario described in Section 3.2.3, i.e. propagating the entire provider profile constitutes a trivial solution that is theoretically applicable for all filtering techniques.
  - *Constrained Propagation via IR*: In the mixed IF/IR scenario, only a small part of the provider profile has to be propagated, the elements of which are based on a non-privacy-preserving IR query.
  - *Partial Propagation via Unlinkable Queries*: In all other cases, the relevant parts of the provider profile have to be retrieved based on user profile data. As described in Section 7.2.2.3, the respective queries are privacy-critical and should therefore be propagated in an unlinkable manner. Depending on the size of the user profile, this approach is rather time-critical.
  - *Partial Propagation via Refinement*: A different approach is based on the fact that the provider entity may obtain any information during the profile propagation process as long as it is information that may be deduced from the recommendations themselves after the filtering process. If, for example, the item i is returned as a recommendation, the prior propagation of the part of the provider profile containing the 100 items most similar to i does not allow the provider to deduce additional information about the user profile. In other words, this strategy refines a large number of provider profile items to a comparatively small number of recommendations without using user profile elements within the respective queries. This strategy is obviously not applicable in the case of completely private result data. In the case of semi-private result data, only single recommendations may be generated during a specific filtering process. It is therefore most suitable in the case of linkable and semi-linkable result data.

Filtering techniques to be applied in our approach should be able to deal with at least one of the latter strategies in addition to the trivial solution of complete propagation.

In the following section we specify filtering techniques meeting these requirements.

9.3 Design & Implementation

In this section, we describe three exemplary filtering techniques that meet the requirements given above. The first filtering technique is not based on profile models and therefore applicable for the use cases "get recommendations" in a Hybrid IF System, "get prediction for item", and "get prediction for user", because these use cases require a comparatively small amount of supplier profile data and therefore may be realized without profile models. The other two filtering techniques are model-based and therefore primarily applicable for the main use case "get recommendations" in a Recommender System.

All filtering techniques are ultimately based on determining the similarity of two single items. It should be noted that the actual item similarity algorithm may be chosen freely without affecting the other parts of the respective filtering technique. In particular, this aspects enables the filter entity to actually preserve the privacy of its sensitive data, because the filter entity may adjust or alter the item similarity algorithm independent of the other entities.

The model-based filtering techniques are used for the information processing stage in the context of the partial use case "announce profile element" as well: The centralized model of candidate users is based on item similarity as well, and therefore the models created via the filtering techniques described below may be used in this context.

9.3.1 Item Similarity Algorithm

As its name implies, the item similarity algorithm ft_1 is directly based on item similarity. The similarity of two items i_1 and i_2 which are contained in a profile is determined by a function $sim(i_1, i_2)$. Following [38], we use a cosine-based similarity function, i.e. given the vectors $\vec{I_1}$ and $\vec{I_2}$ containing the attribute values of i_1 and i_2 respectively, the similarity of these items is defined by Equation 9.1[1].

[1]In the original algorithm, the vectors contain user-related data, which is infeasible in our approach. The algorithm, however, is applicable as long as the vectors contain any kind of comparable data.

$$sim_{ft_1}(i_1, i_2) = cos(\vec{I_1}, \vec{I_2}) \tag{9.1}$$

A prediction of the relevance of an item i is generated by comparing the given item with all items of the user profile and returning the largest similarity, as defined by Equation 9.2. It should be noted that no provider profile items have to be propagated in this case apart from the item i, which is assumed to be given.

$$\begin{aligned} pred_{u,s,ft_1,i} &= max(sim_{ft_1}(i, i_1), .., sim_{ft_1}(i, i_n))\ with \\ \{i_1, .., i_n\} &= PR(u) \end{aligned} \tag{9.2}$$

The top-n recommendations in a Hybrid IF System are generated by by determining the pairwise similarity of all items of the two respective user profiles, and by returning the n most similar items that are not already contained in the user profile of the entity initiating the interaction, as defined by Equation 9.3. In case of small or largely equal profiles, less than n recommendations may be returned.

$$\begin{aligned} & REC_{u,u',ft_1,n} = \\ \{i \in (PR_{u'} \setminus PR_u) | \,\forall\, X \subseteq &(PR_{u'} \setminus PR_u \setminus \{i\}) : \\ |X| < n \quad \vee \quad \exists\, x \in X : pred_{u,u',ft_1,i} &> pred_{u,u',ft_1,x}\} \end{aligned} \tag{9.3}$$

A prediction of the similarity of a user is generated by determining the pairwise similarity of all items of the two respective profiles, and by returning the average similarity of items based on the pairwise similarities, as defined by Equation 9.4.

$$\begin{aligned} pred_{u,u',ft_1,u'} &= \sum_{r=1}^{m} \sum_{s=1}^{n} sim_{ft_1}(i_r, j_s) \ \ with \\ \{i_1, .., i_m\} &= PR(u)\ and \\ \{j_1, .., j_n\} &= PR(u') \end{aligned} \tag{9.4}$$

This filtering technique is applicable in our approach because the user profile data may be propagated independent of and prior to the supplier profile data. The supplier profile data is propagated completely in the Matchmaker System-related use cases, which is feasible because in there cases the supplier represents a second user with a comparatively small profile, and it does not have to be propagated at all in the use case "get prediction for item", because no additional supplier profile data is required in this case.

Thus, the requirements of user privacy and provider privacy are addressed adequately. As the filtering technique is not based on profile models, filter privacy is addressed adequately as well, because no other entity obtains any information about the filtering algorithm itself. The requirement of quality is met because the output of this filtering technique in a Privacy-Preserving Information Filtering context is the same as its output in a regular IF context. The requirement of broadness is met because the filtering technique is not domain-specific. The requirement of performance is met because the same algorithms are used as in the context of a regular IF system.

To summarize, the item similarity algorithm may be applied in our approach for Privacy-Preserving Information Filtering. As its implementation is straightforward, details are omitted here.

9.3.2 Item-based Top-N Recommendation Algorithm

In the following, we use the filtering technique described in [38] as an exemplary representative for the class of non-collaborative, non-learning-based filtering techniques.

The item-based top-N recommendation algorithm ft_2 is based on a provider model created during the information processing stage. The k items with highest similarity values with regard to i_2 are contained in the set TOP_{i_2}, excluding i_2 itself. Similarity is determined via the similarity function introduced above. The provider model is constituted by a matrix M containing item similarities according to Equation 9.5 (for performance reasons, all but the k highest values are set to zero for each column of the matrix).

$$\begin{aligned} m_{I_p,ft_1} &= M \ with \\ M_{i_1,i_2} &= sim_{ft_2}(i_1,i_2) \cdot |\{i_1\} \cap TOP_{i_2}| \end{aligned} \tag{9.5}$$

The items contained in the user profile are modeled as a vector $\vec{U}$ with non-zero values for items contained in the user profile according to Equation 9.6.

$$\begin{aligned} m_{I_u,ft_1} &= \vec{U} \ with \\ \vec{U_i} &= |\{i\} \cap PR_U| \end{aligned} \tag{9.6}$$

Recommendations are generated by obtaining the vector $\vec{x} = M \cdot \vec{U}$, which contains sums of similarity values for all items i: The value $\vec{x}_i$ is the sum of the similarity values of item i with all items contained in the user profile. Based on $\vec{x}$, the N items with the highest values that are not already contained in the user profile are returned as recommendations.

This filtering technique is applicable in our approach because the user profile data may be propagated independent of and prior to the provider profile data. The provider profile data may be propagated as follows:

- *Complete Propagation*: Propagating the entire model is obviously possible. Its size is in $O(I)$ as long as k is chosen independent of $|I|$, which is optimal (a complete model should at least contain information about all profile elements, and thus its size cannot be smaller than $O(I)$) but still infeasible for large provider profiles.

- *Constrained Propagation via IR*: If only a part of the provider profile is selected in a non-privacy-preserving way as described above, only the rows of the matrix M containing the respective elements have to be propagated, and the complete matrix may be reconstructed afterwards by using zero values in the remaining rows. Thus, the propagated part of the provider model may be reduced to a manageable size.

- *Partial Propagation via unlinkable queries*: The values of the columns of the matrix M corresponding to zero values in the vector $\vec{U}$ do not contribute to the sums contained in the vector $\vec{x}$. Therefore, the complete matrix may be reconstructed after obtaining single columns of the matrix M via unlinkable queries by using zero values in the remaining columns. Thus, this approach is applicable for this filtering technique as well.

- *Partial Propagation Via Refinement*: The filtering technique is not suitable for this strategy, because the provider profile cannot be reconstructed or traversed iteratively.

While the requirements of user privacy and provider privacy are thus addressed adequately, the filtering technique has to be adapted in order to address the requirement of filter privacy: If the filtering algorithm is regarded as sensitive data, the provider profile model has to be protected because it contains similarity values results of the part of the filtering algorithm used during the information processing stage. By accessing these similarity values, the provider entity could be able to reconstruct the item similarity algorithm, or it could use the data in order to carry out filtering processes by itself. Therefore, the provider profile model has to be encrypted by the filter entity before it is propagated to the provider entity for storage. The model cannot be encrypted as a whole, as this would prevent the propagation of parts of the models in the context of constrained or partial propagation. Therefore, the rows of the matrix M are encrypted separately, without changing their order, so that the provider is still able to provide the requested data. Nevertheless, the model should be re-encrypted completely whenever items are added or removed, because all other approaches, such as only encrypting altered elements of the matrix M individually, would allow the provider entity to obtain additional information. The requirement of quality is met because the output of this filtering technique in a Privacy-Preserving Information Filtering context is the same as its output in a regular IF context. The requirement of broadness is met because the filtering technique is not domain-specific. The requirement of performance is met in the approaches based on constrained or partial propagation because the same algorithms are used as in the context of a regular IF system, and the additional operations required for encryption and decryption of the model do not change the overall complexity class.

To summarize, the item-based top-N recommendation algorithm may be applied in our approach for Privacy-Preserving Information Filtering. As its implementation is straightforward, details are omitted here.

9.3.3 Hierarchical Clustering-based Algorithm

As the item-based top-N recommendation algorithm is not suitable for the provider profile propagation strategy of partial propagation via refinement, we introduce a different algorithm which utilizes this strategy. Taken together, the two algorithms cover all propagation strategies. Hierarchical clustering algorithms are ideally suited for approaches determining recommendations in an iterative manner, because of the structure of the underlying model. We describe a hierarchical agglomerative clustering-based algorithm as a representative of this class of algorithms.

As the name implies, hierarchical clustering algorithms create a hierarchy of clusters containing similar elements, resulting in a tree representing the element set, which in our case is the provider profile. The leaves of this tree are single profile elements as clusters of size one. In our case, the algorithm ft_3 creates a tree via *single-link clustering*, i.e. by iteratively merging the two most similar clusters, where cluster similarity is defined via item similarity, and thus the two clusters containing the two most similar items are considered to be most similar[2]. These items are additionally defined as the cluster representatives. The process is repeated until all items are merged

[2]Other clustering approaches, such as complete-link clustering or average-link clustering, could also be used here. The only differ with regard to the definition of cluster similarity.

into one single cluster representing the root of the tree. An example is given in Appendix C.

Recommendations are determined by iterating through the tree, from the root upwards, by selecting in each single step the cluster whose representative is most similar to a given user profile element (or a group of user profile elements). Once a cluster of sufficiently small size is reached, its elements may be used either directly as recommendations, or as candidates for recommendations from which, via additional similarity measurements, the actual recommendations are determined.

This filtering technique is applicable in our approach because the user profile data may be propagated independent of and prior to the provider profile data. The provider profile data may be propagated as follows:

- *Complete Propagation*: Propagating the entire model is obviously possible. Its size is in O(I), as there are $2 \cdot |I| - 1$ clusters (or somewhat less if only clusters of at least size k are maintained). Again, this value is optimal but still infeasible for large provider profiles.

- *Constrained Propagation via IR*: If only a part of the provider profile is selected in a non-privacy-preserving way as described above, either the clustering process has to be carried out for the constrained profile, which may be infeasible due to time constraints, or a model based on the complete tree has to be used, whereas items not contained in the constrained profile are simply ignored, which may lead to results of lower quality. Therefore, the filtering technique is less suitable for this strategy.

- *Partial Propagation via unlinkable queries*: Unless combined with the following strategy, the filtering technique is not suitable for this strategy either, because the model is based on clusters of items rather than on single items. Otherwise, the only way to use this strategy would be to carry out at least part of the filtering process at the provider side, which is contrary to our view of the filter entity as independent of other entities.

- *Partial Propagation Via Refinement*: The filtering technique is ideally suitable for this strategy, because it may be carried out iteratively by requesting the respective cluster representatives via separate queries. The queries do not reveal any additional information because they do not contain user profile elements, and they refer to clusters that may be reconstructed via the recommendations themselves, as described in the example given in Appendix C.

As in the case of the filtering technique utilizing the item-based top-N recommendation algorithm described above, this filtering technique has to be adapted as well in order to address the requirement of filter privacy. Again, the provider profile model has to be encrypted by the filter entity before it is propagated to the provider entity for storage. To facilitate constrained or partial propagation of the profile, each cluster has to be encrypted separately. Over time, an honest-but-curious provider may be able to deduce the contents of specific clusters based on the generated recommendations. As this knowledge would still not enable the provider to carry out the filtering process by itself, we consider it to be less privacy-critical. In any case, a concerned filter entity may offset this threat by re-encrypting the complete model periodically, a task which is necessary anyway whenever items are added or removed to the profile. Complete filter privacy may only be achieved by combining the two approaches for partial propagation. The requirement of quality is met because the output of this filtering technique in a Privacy-Preserving Information Filtering context is the same as its output in a regular IF context. The requirement of broadness is met because the filtering technique is not domain-specific. The requirement of performance is met because the same algorithms are used as in the context of a regular IF system, and the additional operations required for encryption and decryption of the model do not change the overall complexity class.

To summarize, the hierarchical agglomerative clustering-based algorithm may be applied in our approach for Privacy-Preserving Information Filtering. As its implementation is straightforward, details are omitted here.

9.4 Summary

This chapter describes exemplary filtering techniques that may be used by the Recommender Module and the Matchmaker Module. We briefly motivate the need for providing exemplary filtering techniques (Section 9.1). We discuss the main aspects that have to be considered for filtering techniques to be applied in our approach (Section 9.2). We describe three exemplary filtering techniques, namely an item similarity algorithm (Section 9.3.1), an item-based top-N recommendation algorithm (Section 9.3.2), and a hierarchical agglomerative clustering-based algorithm (Section 9.3.3), and show that they meet all requirements, as summarized in Table 9.1. The following chapter evaluates our approach in general and in the context of the prototypical application described in Section 4.3.

Table 9.1: An overview of the introduced exemplary filtering techniques in relation to the requirements and acceptance aspects of Privacy-Preserving Information Filtering for the use cases realized via the respective filtering technique. A requirement is fully met (indicated by "✓"), partially met (indicated by "○"), or not met at all (indicated by "–"). Acceptance is indicated in an analogous manner. The term "propagation" refers to the propagation of the supplier profile.

	Privacy Requirements			Other Requirements			Acceptance	
	R_u	R_p	R_f	R_{qq}	R_{bb}	R_{pp}	A_u	A_p
item similarity algorithm ft_1								
complete propagation	✓	✓	✓	✓	✓	✓	✓	✓
without propagation	✓	✓	✓	✓	✓	✓	✓	✓
item-based top-N recommendation algorithm ft_2								
complete propagation	✓	✓	✓	✓	✓	–	✓	○
constrained prop. via IR	✓	✓	✓	✓	✓	✓	✓	✓
partial prop. via queries	✓	✓	✓	✓	✓	○	✓	○
partial prop. via refinement	n/a							
hierarchical agglomerative clustering-based algorithm ft_3								
complete propagation	✓	✓	✓	✓	✓	–	✓	○
constrained prop. via IR	✓	✓	✓	○	✓	○	✓	✓
partial prop. via queries	✓	✓	–	✓	✓	✓	✓	○
partial prop. via refinement	✓	✓	○	✓	✓	✓	✓	✓

Chapter 10

Evaluation

This chapter evaluates our approach for Privacy-Preserving Information Filtering. It is structured as follows: Section 10.1 discusses the coverage of the non-functional and functional requirements by our approach and by the implemented application. Section 10.2 compares our approach with approaches based on trusted software. Section 10.3 evaluates the applicability of the functionality specified in our approach and provides usage guidelines. Section 10.4 concludes the chapter with a summary.

10.1 Coverage of Requirements

In this section, we show that our approach meets all functional and non-functional requirements listed in Section 2.3.4. In addition to a theoretical evaluation, we also discuss results obtained from evaluating the prototypical application that we have implemented based on our approach, as described in Section 4.3. It should be noted, though, that most non-functional requirements (such as the requirement of privacy) cannot be quantified easily, and some of them (such as the requirement of broadness) cannot be quantified at all, and therefore we cannot actually measure the extent to which these requirements are met in the prototypical application. With regard to these requirements, we have to rely on the theoretical evaluation.

10.1.1 Functional Requirements

In the following, we review the functional requirements and show that they are actually met by our approach. As shown in Table 10.1, not all required functionality has actually been implemented for the prototypical application, mainly because the application focuses on Recommender System functionality rather than Matchmaker System functionality.

- The system should provide sufficient functionality for realizing the information collection stage, the information processing stage and the information filtering stage of an IF system. This requirement is met by our approach, because it provides for the respective main use cases:
  - The information collection stage is covered by functionality for the main use case “update profile elements”, as described in Section 7.2.2.1.
  - The information processing stage is covered by functionality for the main use case “update profile model”, as described in Section 7.2.2.2.
  - The information filtering stage is covered by functionality for the main use cases “get prediction for item”, “get recommendations”, “get prediction for user”, and “get similar users”, as described in Section 7.2.2.3 and Section 8.2.2.

  The respective functionality has been implemented in the prototypical application, with the exception of functionality for the use cases “get prediction for user” and “get similar users”.

- The system should be able to return all different kinds of result data defined in Section 2.2.1, namely predictions of the relevance of specific items, top-n recommendations of items, predictions of the similarity of specific users, and top-n similar users for a given user. In other words, the system should be able to provide Recommender System functionality as well as Matchmaker System functionality. As described above, the different kinds of result data are covered by the different use cases of the information filtering stage, and thus this requirement is met by our approach, as described in Section 7.2.2.3.

- Regarding filtering techniques, the system should be able to support feature-based approaches as well as collaborative approaches. This requirement is met by our approach, because the Recommender Module utilizes feature-based approaches, while the Matchmaker Module utilizes collaboration-based approaches for the generation of the centralized model. Two exemplary feature-based approaches, both of which have been implemented in the prototypical application, are described in Chapter 9.

Table 10.1: Coverage of the functional requirements of an IF architecture by our approach for PPIF and the implemented prototypical application. A requirement is covered (indicated by "✓") or not covered (indicated by "–").

functional requirement	covered by our approach	implemented in application
information collection stage	✓	✓
information processing stage	✓	✓
information filtering stage	✓	✓
item predictions	✓	✓
recommendations	✓	✓
user similarity predictions	✓	–
similar users	✓	–
feature-based filtering techniques	✓	✓
collab.-based filtering techniques	✓	–

10.1.2 Non-functional Requirements

In the following, we show that in addition to the functional requirements, the non-functional requirements are also actually met by our approach. We also discuss acceptance aspects here because they are closely related to non-functional requirements. As shown in Table 10.2 and Table 10.5, not all specified functionality has actually been implemented for the prototypical application. The part of the functionality that has been implemented, however, is sufficient for evaluating the quantifiable requirements, such as performance.

10.1.2.1 Privacy

The requirement of privacy is mainly met by keeping private data under the control of the respective entity. Other entities may only access the data temporarily, without being able to propagate the data to any other entity or in any other way that would cause the first entity to lose control of the further dissemination of its private data.

This solution is basically realized via mechanisms for communication control, as described in Section 5.2.2.1, which are applied in the interactions of the Recommender Module in order to protect privacy against threats originating from honest-but-curious adversaries, as described in Section 7.2.2. The respective protocols are extended by steps for secure message forwarding in order to protect privacy against threats originating from malicious adversaries, as described in Section 7.3. Communication control and the extended protocols have been implemented in the prototypical application. Communication control has to be based on a trusted environment, which may be realized via a trusted computing infrastructure but is not implemented in the prototypical application. This solution applies to all entities and thus addresses user privacy, provider privacy, and filter privacy.

In parts of protocols where communication control is not applicable or insufficient, the requirement of privacy is additionally addressed via anonymous communication and encryption of private data, as described in the following for the different main abstract entities:

- User privacy: Privacy of the user entity is additionally preserved via anonymous communication in the following areas of our approach:
  - Query data propagation: User profile data has to be propagated to the provider entity in case of large provider profiles. Unlinkability of user and profile elements as well as unlinkability of profile elements among themselves is achieved by propagating single elements via anonymous interactions, as described in Section 7.2.2.3. This is not required for the scenarios of the prototypical application, as described in Section 10.1.2.4, and therefore not implemented.
  - Result data propagation: User profile data may be reconstructed from the result data, which is propagated to the provider entity unless the partial use case "completely private result data" is realized. In case of semi-linkable or semi-private result data, reconstruction is prevented by propagating result data via anonymous interactions, as described in Section 7.2.2.3. The prototypical application realizes the partial use case "completely linkable result data", and therefore the other partial use cases are not implemented in the prototypical application.
  - Determining Potentially Similar Users: The Matchmaker Module determines potentially similar users via a centralized model containing user profile data. As in the case of query data propagation, unlinkability is achieved via anonymous communication, as described in Section 8.2.2.1. As noted above, Matchmaker System functionality is not implemented in the prototypical application.
- Provider privacy: Privacy of the provider entity is additionally preserved via data encryption in the area of query data propagation of our approach: As described above, anonymous communication is used to

protect the user privacy in this context, which implies that communication control cannot be used to protect provider profile data, which is propagated to the user entity within the respective interactions, because controlled agents cannot communicate anonymously. In order to protect the privacy of the provider profile data up to the point in time when the respective user agent may be controlled, the data is encrypted by the provider entity as described in Section 7.2.2.3. As noted above, this is not required for the scenarios of the prototypical application, and therefore not implemented.

- Filter privacy: Privacy of the filter entity is additionally preserved via data encryption in the area of information processing of our approach: As described in Section 7.2.2.2, the provider profile model may allow an honest-but-curious entity to obtain information about the filtering algorithm, or even to carry out subsequent filtering processes by itself. In order to protect the privacy of the filtering algorithm in this context, the model is encrypted by the filter entity as described in Section 9.3 in the context of exemplary filtering techniques. This is implemented in the prototypical application.

Table 10.2: Coverage of the privacy requirements of an IF architecture in our approach for PPIF and the implemented prototypical application. A requirement is covered (indicated by "✓") or not covered (indicated by "–").

non-functional requirement/ solution	covered by our approach	implemented in application
privacy (all entities)		
communication control	✓	✓
extended protocols	✓	✓
trusted environment	✓	–
user privacy		
anonymous communication	✓	–
provider privacy		
data encryption	✓	–
filter privacy		
data encryption	✓	✓

To summarize, the requirement of privacy is met by our approach via a combination of communication control, anonymous communication and data encryption, as shown in Table 10.2.

10.1.2.2 Quality

While the requirement of quality is quantifiable and thus could be evaluated practically e.g. by comparing the quality of recommendations provided by the prototypical application with the quality of recommendations provided by a regular IR system within the same domain, it is actually sufficient to evaluate this requirement theoretically: As described in Chapter 9, the algorithms the filtering techniques are based on are the same algorithms that are used in a regular IF system, and therefore the quality of the result data does not change compared to a regular IF system.

10.1.2.3 Broadness

The requirement of broadness implies that the architecture should not be restricted to single information domains, specific filtering techniques, or specific persistent storage mechanisms. It is met by our approach due to the fact that the specification of functionality does not bring about any restrictions of this kind. In particular, the Infrastructure Module, the Recommender Module and the Matchmaker Module are designed in a domain-independent way. The TPMAS Module described in Chapter 6 provides a transparent persistence interface which allows the use of arbitrary persistent storage mechanisms. Domain-specific filtering techniques may be used for domain-specific applications based on our approach, but, as shown in Chapter 9, domain-independent filtering techniques may be applied effectively.

10.1.2.4 Performance

The requirement of performance is theoretically met by our approach, because the introduced protocols do not change the computational complexity class or the communication complexity class with regard to profile data and results, compared to a regular IF approach, as the amount of information propagated and processed in each case is in the same class. However, as exemplary shown in Table 10.3 for a typical use case, there is a substantial amount of additional interactions and computations which are constant with regard to profile data and results, but obviously affect the performance in practice considerably.

Therefore, we also evaluate performance based on the implement prototypical application described in Section 4.3, i.e. the Smart Event Assistant. In this application, recommendations are provided in two ways, based on an item-based top-N recommendation algorithm: A push service delivers new recommendations to the user in regular intervals (e.g. once per day) via email or SMS. Because the user is not online during these interactions, they are

Table 10.3: The performance of a typical privacy-preserving filtering process realizing the use case "get recommendations" with linkable result data, assuming honest-but-curious participants, compared to the performance of the respective non-privacy-preserving process.

	# main computations		# main interactions	
	non-privacy-preserving	privacy-preserving	non-privacy-preserving	privacy-preserving
Phase *I*				
create agent	–	2	–	2
restrict comm.	–	1	–	3
propagation PR_U	–	–	1	3
Phase *II*				
restrict comm.	–	2	–	6
propagation PR_S	–	–	1	3
Phase *III*				
filtering algorithm	1	1	–	–
propagation RES	–	–	1	3
terminate agent	–	2	–	6

less critical with regard to performance and the protracted duration of the information filtering process is acceptable in this case. Recommendations generated for the intelligent day planner service, however, have to be delivered with very little latency because the process is triggered by the user, who expects to receive results promptly. In this scenario, the overall performance is substantially improved by setting up the relay agent and the TFE agent offline, i.e. prior to the user's request, and by the fact that the interactions are based on the mixed IR/IF scenario, and thus the relevant part of the provider profile may be propagated in an efficient manner: Because the user is only interested in items, such as movies, available within a certain time period and related to specific locations, such as screenings at cinemas in a specific city, the relevant part of the provider profile is usually small enough to be propagated entirely. As these additional parameters are not seen as privacy-critical (because they are not based on the user profile, but rather constitute a short-term information need), the relevant part of the provider profile may be propagated as a whole, with no need for more complex interactions. Thus, even a higher number of elements may be retrieved in an efficient manner, because only a single query and a single interaction iteration is required.

Taken together, these improvements result in a filtering process that, according to our evaluation, takes about three times as long as the respective non-privacy-preserving filtering process, which we regard as an acceptable trade-off for the increased level of privacy. Table 10.4 shows the results of the performance evaluation in more detail.

Table 10.4: The performance of typical privacy-preserving filtering processes in the Smart Event Assistant, compared to the performance of the respective non-privacy-preserving process. In the non-privacy-preserving version, a filter agent retrieves the profiles directly from a database and propagates the result to a provider agent.

	push scenario		day planning scenario	
	non-privacy-preserving	privacy-preserving	non-privacy-preserving	privacy-preserving
profile size (retrieved items/total amount of items)				
user profile	25/25		25/25	
provider profile	125/10,000		500/10,000	
elapsed time in filtering process (in seconds)				
agent creation etc.	n/a	2.2	n/a	offline
database access	0.2	0.5	0.4	0.4
profile propagation	n/a	0.8	n/a	0.3
filtering algorithm	0.2	0.2	0.2	0.2
result propagation	0.1	1.1	0.1	1.1
overall time	0.5	4.8	0.7	2.0

To summarize, the requirement of performance is met by our approach for different scenarios, as shown in Table 10.5 together with the other non-privacy-related requirements.

10.1.2.5 User Acceptance

With regard to user acceptance, we consider our approach to be sufficiently acceptable in terms of usability, mainly because the user interactions are largely the same as in a regular IF system. Compared to other conceivable approaches for Privacy-Preserving Information Filtering, such as centralized approaches based on trusted computing, the user is expected to trust an application based on our approach to a higher degree, because the concept of a personal user agent containing and protecting personal data may be immediately more comprehensible than the concept of privacy protection via trusted

Table 10.5: Coverage of the requirements of quality, broadness, and performance in our approach for PPIF and the implemented prototypical application. A requirement is covered (indicated by "✓") or not covered (indicated by "–").

non-functional requirement/ solution/ scenario	covered by our approach	implemented in application
quality		
standard IF filtering techniques	✓	✓
broadness		
transparent persistence	✓	✓
domain-independent protocols	✓	✓
domain-independent filtering techniques	✓	✓
performance		
use cases in general	○	–
recommendations for offline user	✓	✓
mixed IR/IF scenario	✓	✓

computing technology. While it should be noted that our solution is based on a trusted environment which is likely to be realized via trusted computing as well, it may still be more acceptable especially if the trusted environment is provided by an independent party and used for various additional tasks.

10.1.2.6 Provider Acceptance

We consider provider acceptance to be the greatest challenge of our approach. As long as users continue to accept and use non-privacy-preserving Recommender Systems and Matchmaker Systems, providers are not likely to embrace approaches for Privacy-Preserving Information Filtering mainly because they obtain less information about users, and have to provide additional resources. However, as discussed in Section 2.2.3, this may change in case of more complex applications, or in case of applications for more sensitive domains, such as healthcare-related or financial Recommender Systems. Ultimately, in order to acquire and retain large numbers of users for these kinds of applications, providers will have to trade-off information about users for user acceptance, and thus are expected to accept approaches for PPIF perforce at least in specific scenarios.

10.2 Trusted Software Approaches

As described in Section 4.2.1, our solution is based on a trusted environment in a Multi-Agent System technology context, which implies that the problem of malicious hosts has to be addressed. As discussed in Section 3.3.2, a trusted computing infrastructure is the only viable technology-based solution for this problem, and therefore our approach is based on a trusted computing infrastructure. Furthermore, as described in Section 3.1.3, trusted computing has been suggested as a Privacy-Enhancing Technology and could be applied to a regular IF system in a rather straightforward way, resulting in system realizing Privacy-Preserving Information Filtering via the trusted software approach. In this section, we discuss the drawbacks of approaches based on trusted software, and show that our approach based on a trusted environment, while somewhat more complex, is in fact more suitable with regard to the requirements for Privacy-Preserving Information Filtering.

10.2.1 Broadness

The main problem of a solution for PPIF based on trusted software is its lack of broadness: a software is trusted by a user in the context of trusted computing when the user is able to verify, via examination, that the software works as specified, and when the user is able to verify, via remote attestation, that the instance of the software deployed by the provider actually matches the examined software and is actually running within a trusted computing infrastructure. Both aspects are problematic because they have to be carried out for each single version of each single system in which the user participates. Typically, the user himself does not have the knowledge or the resources to examine software, and thus has to rely on third parties for this task. In any case, the examination has to be carried out whenever some part of the respective software, including the filtering algorithms in the context of IF, is patched, upgraded or replaced.

In contrast, in a solution based on a trusted environment, only a comparatively small part of the overall system has to be examined and attested. Other parts of the system, such as filtering algorithms or the protocols used in the interactions, may be changed without changing the trusted environment itself.

10.2.2 Provider Acceptance

Furthermore, a solution based on a trusted environment may be more acceptable for providers basically because the trusted environment could be used for various other tasks and applications involving mobile agents: As discussed in Section 11.2, it could actually be used to realize applications from most areas of related work. Furthermore, as all applications based on mobile agents ultimately require a trusted environment, it seems to be realistic to assume that there is a greater chance that such a trusted environment is actually realized, as compared to the chance that trusted software for a large number of very specific tasks is actually realized.

10.3 Usage Guidelines

In this section, we evaluate the main use cases of our approach including partial use cases with regard to applicability. While the provided functionality may be combined arbitrarily, not all combinations are equally viable especially with regard to performance, nor are they all equally useful. Therefore, in a real-world system based on our approach, we do not expect all combinations to be available. In any case, even privacy-aware users are probably not interested in having to decide whether the received result data should be e.g. semi-linkable or semi-private. We give the following usage guidelines, based on typical scenarios in Recommender Systems and Matchmaker Systems:

- Recommender System; use case "get recommendations" based on the complete provider profile: This use case is generally not that time-critical with regard to performance, because it addresses a long-term information need of a user and is based on relatively static profiles (in the sense that profile updates are expected to take place in intervals that are much longer (e.g. several hours) than the time required to carry out a filtering process (e.g. a few seconds). Subsequently, the use case is not based on immediate user input and can be realized without human user interaction. At the same time, the result data generated in this use case would allow honest-but-curious participants to deduce information about the user profile to a larger extent than in the context of other use cases, and therefore the result data is highly privacy-critical. We therefore suggest to use the protocol for semi-private or completely private result data in this case, and propagate a partial provider profile based on unlinkable queries.

- Recommender System; use case "get recommendations" based on a constrained provider profile that represents the result of a query in the mixed IR/IF scenario: This use case is more time-critical because the query based on which the constrained profile is determined is expected to have been created via interaction with a human user waiting for results. At the same time, the result data may not be as privacy-critical as in the scenario described above, basically because it may reflect the user profile to a lesser degree, depending on the query and the number of results obtained. If, for example, the constrained profile only contains items that would never have been suggested as recommendations if the complete profile had been used, the result data may not allow other entities to correctly deduce information about the user profile. We therefore suggest to use the protocol for completely linkable or semi-linkable result data in this case, and propagate the entire constrained provider profile.

- Matchmaker System; use case “get similar users”: This use case again addresses a long-term information need of a user that is based on relatively static profiles, and is therefore not considered as time-critical with regard to performance. Furthermore, it is also not highly privacy-critical, again because the result data may not reflect the respective user profiles closely. Users may also trust other users to a larger degree than provider, i.e. other users are usually expected to act non-maliciously. We therefore suggest to use the protocol for completely linkable or semi-linkable result data in this case, and propagate the entire supplier profile.

All other combinations of use cases occur less often in real-world IF systems and are therefore omitted here.

10.4 Summary

This chapter evaluates our approach for Privacy-Preserving Information Filtering. We discuss the coverage of the non-functional and functional requirements by our approach and by the implemented application (Section 10.1), and show that all requirements are addressed adequately. We compare our approach with approaches based on trusted software (Section 10.2), and show that approaches based on a trusted environment meet the requirements to a higher degree. Finally, we evaluate the applicability of the functionality provided by our approach (Section 10.3) and provide usage guidelines for applications based on our approach. The following chapter concludes this work and outlines future directions of research.

Chapter 11

Conclusion & Outlook

This work describes an agent-based approach for Privacy-Preserving Information Filtering. This approach utilizes mechanisms that allow entities to control the communication capabilities of other entities, combined with an infrastructure for anonymous communication and data encryption. Based on these building blocks, a trusted environment providing functionality for the various stages of Information Filtering is realized. We describe how this functionality may be used for obtaining multilateral privacy in the context of Information Filtering, i.e. privacy of all participating entities. The approach provides functionality for privacy-preserving Recommender Systems, distributed Matchmaker Systems, and combinations thereof. It utilizes fundamental features of agents such as autonomy, adaptability and the ability to communicate, which make agents uniquely suitable for representing the entities participating in Privacy-Preserving Information Filtering. As a proof of concept, an in order to be able to evaluate the approach practically, especially with regard to performance, we have implemented the Smart Event Assistant as a prototypical application supporting users in planning entertainment-related activities in a privacy-preserving manner.

This chapter concludes the work by discussing the applicability of the approach in large-scale real-world applications (Section 11.1) and by discussing directions for further research and development (Section 11.2).

11.1 Applicability

If our approach is to be successfully applied in a large scale real-world Recommender System or Matchmaker System, the following aspects have to be addressed:

- An industrial-strength MAS technology is required as a robust foundation for the implemented functionality. Application providers are only expected to accept the use of MAS technology in the context of a commercial application if it achieves a level of maturity with regard to stability, required maintenance effort, support and ease of development that is comparable to other technologies. Because most current MAS technologies have been developed within a research context, they have not been widely used in commercial applications yet.

- Feedback obtained from users of the Smart Event Assistant indicates that most users are indifferent to privacy in the context of entertainment-related personal information. Therefore, in order to achieve a high degree of user acceptance especially in view of expected minor performance trade-offs, the application should probably focus on information of a more privacy-critical domain, such as healthcare-related or financial information.

- Finally, in order to further increase user acceptance, the personal agent containing and controlling the private user data should be physically associated with the respective user: Rather than operating on a platform running on a server supplied by the application provider, the personal agent should operate on a platform running on a device owned by the user, even though the former approach is also feasible if the agent operates within a trusted environment at all times. The latter approach is probably realized best by utilizing mobile devices, because a mobile device is available to the respective user in more situations than e.g. his PC. This approach requires the respective MAS technology to be deployed on a number of different mobile devices with different operating systems, which is not possible with current MAS technologies.

To summarize, we consider the degree of maturity of current MAS technologies to be the biggest obstacle with regard to a successful real-world application based on our approach.

11.2 Future Work

We envision two main areas of future work, namely the further implementation of our approach and possible generalizations of our approach addressing related problems:

- As noted in Section 10.1, not all functionality described in this work has actually been implemented. While the functionality that has been implemented is sufficient as proof of concept, in a complete framework for Privacy-Preserving Information Filtering based on our approach the trusted environment has to be realized via mechanisms for trusted computing, which remains future work. Matchmaker System functionality and an infrastructure for anonymous communication has to be implemented as well. Finally, in our implementation we have mainly focused on enabling agents to carry out the protocols as specified. In the optimal case, however, agents should additionally be able to actually analyze the data obtained via the interactions in order to detect possible threats originating from malicious participants.

- Our approach of realizing multilateral privacy via communication control in a trusted environment could be applied to other problems related to PPIF as well. As discussed in Chapter 3, Secure Multi-Party Computation protocols have been suggested for a number of tasks in various Privacy-Enhancing Technologies and Privacy-Preserving Technologies. in these cases, the respective computation could be carried out by a controlled agent within a trusted environment, which is feasible with regard to communication complexity unless the number of parties is large.

We consider the latter area to be especially relevant, because the respective solutions would further highlight the broad applicability of our approach, as opposed to approaches based on trusted software.

Appendix A

Specification of Ontologies, Roles and Interactions

This appendix provides tables and diagrams containing the formal specification of the components of our approach for Privacy-Preserving Information Filtering.

A.1 Basic Infrastructure

This section contains specification related to the Infrastructure Module described in Chapter 5.

A.1.1 Ontologies

Figure A.1 shows the categories of the ontology "Communication Rules". Figure A.2 shows the categories of the ontology "Anonymity".

A.1.2 Interactions

The basic interactions of the roles participating in communication control are specified in the following tables: Table A.1 describes the interaction RestrictCommunication. Table A.2 describes the interaction CheckRule. Table A.3 describes the interaction ActivateRule. Table A.4 describes the interaction AcquireConsent.

Usually the analysis phase and thus the specification of interactions abstracts from agents and platform configurations. However, in this case it is necessary to refer to these concepts because of the reflective nature of the

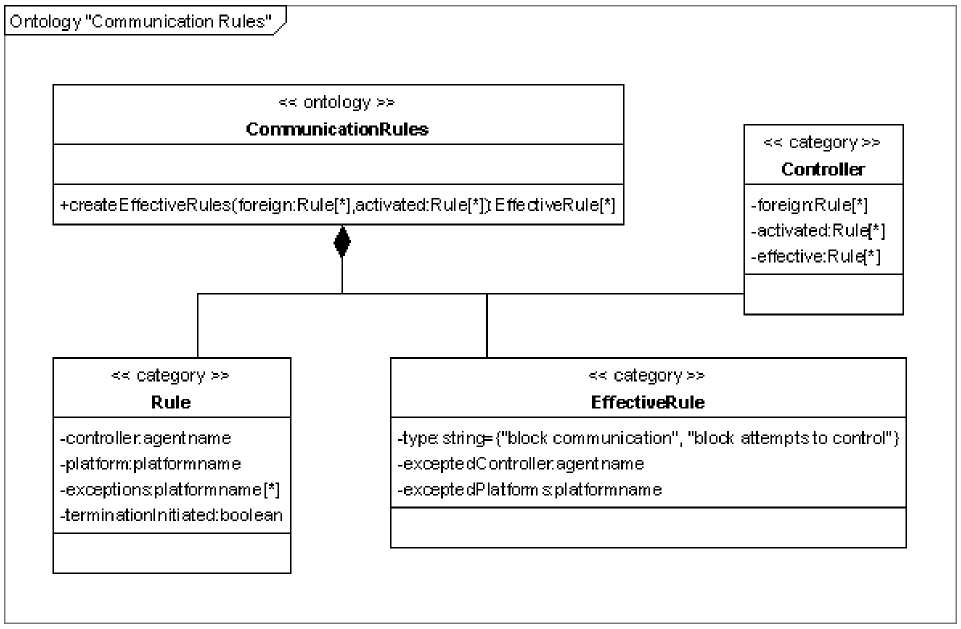

Ontology "Communication Rules"
<< ontology >>
CommunicationRules
+createEffectiveRules(foreign:Rule[*],activated:Rule[*]):EffectiveRule[*]
<< category >>
Controller
-foreign:Rule[*]
-activated:Rule[*]
-effective:Rule[*]
<< category >>
Rule
-controller:agentname
-platform:platformname
-exceptions:platformname[*]
-terminationInitiated:boolean
<< category >>
EffectiveRule
-type:string={"block communication", "block attempts to control"}
-exceptedController:agentname
-exceptedPlatforms:platformname

Figure A.1: The categories of the ontology "Communication Rules".

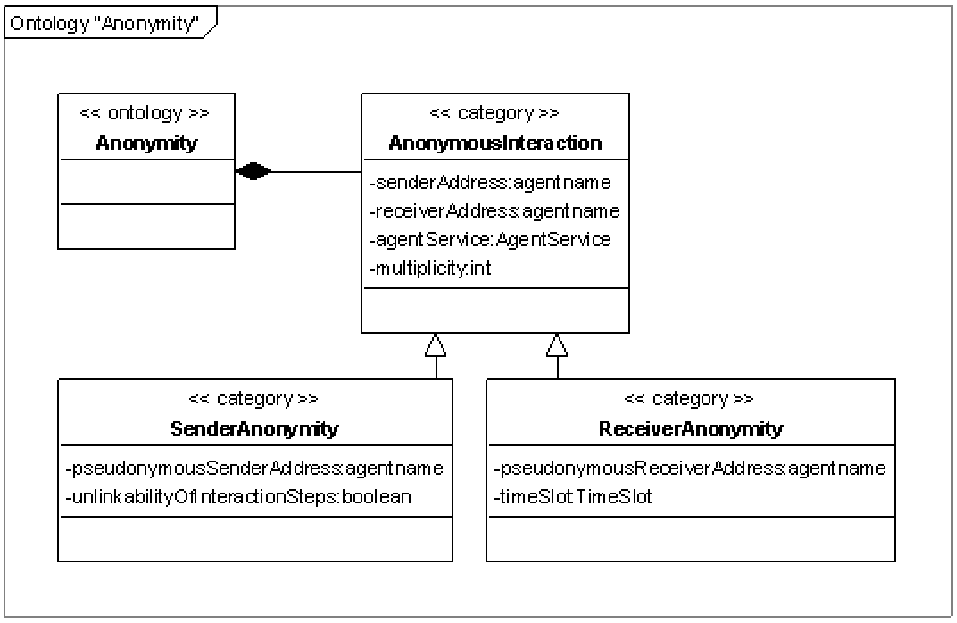


Figure A.2: The categories of the ontology "Anonymity".

interactions. Therefore, the terms “local” and “remote” refer to platform configurations with regard to agents realizing the respective roles.

Table A.1: The interaction RestrictCommunication.

Initiator:	AgentRole
Partner:	SupervisorRole (local)
Input:	Platforms to be controlled
Output:	Status information
Description:	Interaction for restricting communication. As a result, the given platforms are controlled as far as possible.

Table A.2: The interaction CheckRule.

Initiator:	SupervisorRole
Partner:	SupervisorRole (remote)
Input:	Rule to be activated
Output:	Status Information (indicating whether all agents have consented), further platforms to be controlled
Description:	Interaction for checking whether a rule may be activated.

Table A.3: The interaction ActivateRule.

Initiator:	SupervisorRole
Partner:	SupervisorRole (remote)
Input:	Rule to be activated
Output:	Status information
Description:	Interaction for activating a rule.

Table A.4: The interaction AcquireConsent.

Initiator:	SupervisorRole
Partner:	ControllableAgentRole (local)
Input:	Rule to be activated
Output:	Consent or rejection
Description:	Interaction for acquiring consent to controlling the respective agent.

Interactions for revoking control are specified in the following tables: Table A.5 describes the interaction RevokeControl. Table A.6 describes the interaction RevokeRule. An additional interaction for cascading control is specified in Table A.7, which describes the interaction InformAboutCascadingControl.

Table A.5: The interaction RevokeControl.

Initiator:	AgentRole
Partner:	SupervisorRole (local)
Input:	Platforms that are no longer to be controlled
Output:	Status information
Description:	Interaction for revoking control. As a result, the effective control of platforms is revoked as far as possible

Table A.6: The interaction RevokeRule.

Initiator:	SupervisorRole
Partner:	SupervisorRole (remote)
Input:	Rule to be revoked
Output:	Status information
Description:	Interaction for revoking a rule. As a result, the rule is revoked if possible.

Table A.7: The interaction InformAboutCascadingControl.

Initiator:	SupervisorRole
Partner:	ControllableAgentRole (local)
Input:	Rule to be activated
Output:	Status information
Description:	Interaction for notifying an agent on a controlled platform with regard to cascading control.

Interactions related to additional management functionality are specified in the following tables: Table A.8 describes the interaction RevokeControlAndTerminate. Table A.9 describes the interaction RevokeRuleAndTerminate. Table A.10 describes the interaction InformAboutTermination. Table A.11 describes the interaction RequestControl.

An interaction related to anonymous communication is specified in Table A.12, which describes the interaction SetupAnonymizer.

Table A.8: The interaction RevokeControlAndTerminate.

Initiator:	AgentRole
Partner:	SupervisorRole (local)
Input:	Platforms that are no longer to to be controlled
Output:	Status information
Description:	Interaction for revoking control and terminating the respective platforms. As a result, the effective control of the platforms is revoked as far as possible, and agents on the platforms are terminated as far as possible.

Table A.9: The interaction RevokeRuleAndTerminate.

Initiator:	SupervisorRole
Partner:	SupervisorRole (remote)
Input:	Rule to be revoked
Output:	Status information
Description:	Interaction for revoking a rule and terminating the respective platform. As a result, the rule is revoked if possible, and the platform is terminated or marked for termination.

Table A.10: The interaction InformAboutTermination.

Initiator:	SupervisorRole
Partner:	SupervisorRole (remote)
Input:	Terminated platform
Output:	Status information
Description:	Interaction for informing about the termination of a platform. As a result, the respective foreign rules are updated by removing rules regarding the terminated platform.

Table A.11: The interaction RequestControl.

Initiator:	AgentRole (local)
Partner:	AgentRole (remote)
Input:	Request for control of a platform
Output:	Status information
Description:	Interaction for requesting control of a platform. An agent may request control in order to be able to proceed with a specific protocol.

Table A.12: The interaction SetupAnonymizer.

Initiator:	AgentRole
Partner:	AnonymizerRole
Input:	Information required for establishing sender and/or receiver anonymity
Output:	Status information
Description:	Interaction for setting up an anonymizer. As a result, the anonymizer is ready for anonymizing communication.

A.1.3 Role Model

Table A.14 describes the role schema for the SupervisorRole. Table A.15 describes the role schema for the ControllableAgentRole. Table A.13 describes the role schema for the AnonymizerRole.

Table A.13: The role schema for the AnonymizerRole.

Description:	Provides functionality for enabling anonymous communication
Protocols: *Activities:*	$\mathsf{SetupAnonymizer}_{R,P}$ $\underline{\underline{\underline{\mathsf{SetupAnonymizerInternally}}}}$
Lifeness:	$\textsc{ar}_1$
$\textsc{ar}_1 = (\mathsf{SetupAnonymizer}_R.\underline{\underline{\underline{\mathsf{SetupAnonymizerInternally}}}}.\mathsf{SetupAnonymizer}_P)^\omega$	

Table A.14: The role schema for the SUPERVISORROLE.

Description:	Provides functionality for controlling the communication capabilities of agents
Protocols: *Activities:*	$\text{RestrictCommunication}_{R,P}$, $\text{CheckRule}_{R,P,I}$, $\text{ActivateRule}_{R,P,I}$, $\text{AcquireConsent}_{I}$, $\text{RevokeControl}_{R,P}$, $\text{RevokeRule}_{R,P,I}$, $\text{TerminateAgents}_{I}$, $\text{RevokeControlAndTerminate}_{R,P}$, $\text{RevokeRuleAndTerminate}_{R,P,I}$, $\text{InformAboutTermination}_{R,P,I}$ $\underline{\underline{\text{CheckRules}}}$, $\underline{\underline{\text{AddActivatedRule}}}$, $\underline{\underline{\text{DetermineEffectiveRule}}}$, $\underline{\underline{\text{CheckForeignRules}}}$, $\underline{\underline{\text{RemoveForeignRules}}}$, $\underline{\underline{\text{RemoveActivatedRule}}}$
Lifeness:	CR_1, CR_2, CR_3, CR_4, CR_5, CR_6, CR_7, CR_8
$\text{CR}_1 = (\text{RestrictCommunication}_R.\underline{\underline{\text{CheckRules}}}.\text{CheckRule}_I{*}.\text{ActivateRule}_I{*}.\text{RestrictCommunication}_P)^\omega$	
$\text{CR}_2 = (\text{CheckRule}_R.(\text{AcquireConsent}_I{*} \parallel \text{InformAboutCascadingControl}_I{*}).\text{CheckRule}_P)^\omega$	
$\text{CR}_3 = (\text{ActivateRule}_R.\underline{\underline{\text{AddActivatedRule}}}.\underline{\underline{\text{DetermineEffectiveRule}}}.\text{ActivateRule}_P)^\omega$	
$\text{CR}_4 = (\text{RevokeControl}_R.\underline{\underline{\text{CheckForeignRules}}}.\text{RevokeRule}_I{*}.\underline{\underline{\text{RemoveForeignRules}}}{*}.\text{RevokeControl}_P)^\omega$	
$\text{CR}_5 = (\text{RevokeRule}_R.\underline{\underline{\text{RemoveActivatedRule}}}.\underline{\underline{\text{DetermineEffectiveRule}}}.\text{TerminateAgents}_I{*}.\text{RevokeRule}_P)^\omega$	
$\text{CR}_6 = (\text{RevokeControlAndTerminate}_R.\underline{\underline{\text{CheckForeignRules}}}.\text{RevokeRuleAndTerminate}_I{*}.\underline{\underline{\text{RemoveForeignRules}}}{*}.\text{RevokeControlAndTerminate}_P)^\omega$	
$\text{CR}_7 = (\text{RevokeRuleAndTerminate}_R.\underline{\underline{\text{RemoveActivatedRule}}}.\underline{\underline{\text{DetermineEffectiveRule}}}.(\text{TerminateAgents}_I \parallel \text{InformAboutTermination}_I{*}).\text{RevokeRuleAndTerminate}_P)^\omega$	
$\text{CR}_8 = (\text{InformAboutTermination}_R.\underline{\underline{\text{RemoveActivatedRule}}}.\underline{\underline{\text{DetermineEffectiveRule}}}.(\text{TerminateAgents}_I \parallel \text{InformAboutTermination}_I{*}).\text{InformAboutTermination}_P)^\omega$	

Table A.15: The role schema for the CONTROLLABLEAGENTROLE.

Description:	Provides functionality for agents on controllable platforms
Protocols: *Activities:*	$\text{AcquireConsent}_{R,P}$ $\underline{\underline{\text{CheckConsent}}}$
Lifeness:	CAR_1
$\text{CAR}_1 = (\text{AcquireConsent}_R.\underline{\underline{\text{CheckConsent}}}_I{*}.\text{AcquireConsent}_P)^\omega$	

A.2 Transparent Persistence

This section contains specification related to the TPMAS Module described in Chapter 5.

A.2.1 Ontologies

Figure A.3 shows the categories of the ontology "Transparent Persistence". Figure A.4 shows the categories of the ontology "Query Construct".

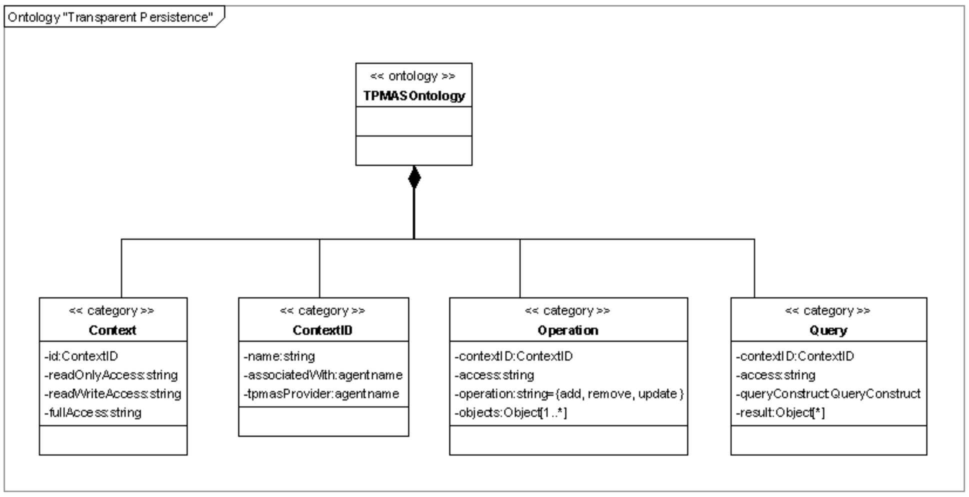


Figure A.3: Diagram for the categories of the ontology "Transparent Persistence".

A.2.2 Interactions

The interactions of the roles participating in the TPMASModule are specified in the following tables: Table A.16 describes the interaction CreateContext. Table A.17 describes the interaction TerminateContext. Table A.18 describes the interaction ModifyObjects. Table A.19 describes the interaction RetrieveObjects.

A.2.3 Role Model

Table A.20 describes the role schema for the TPMASProviderRole.

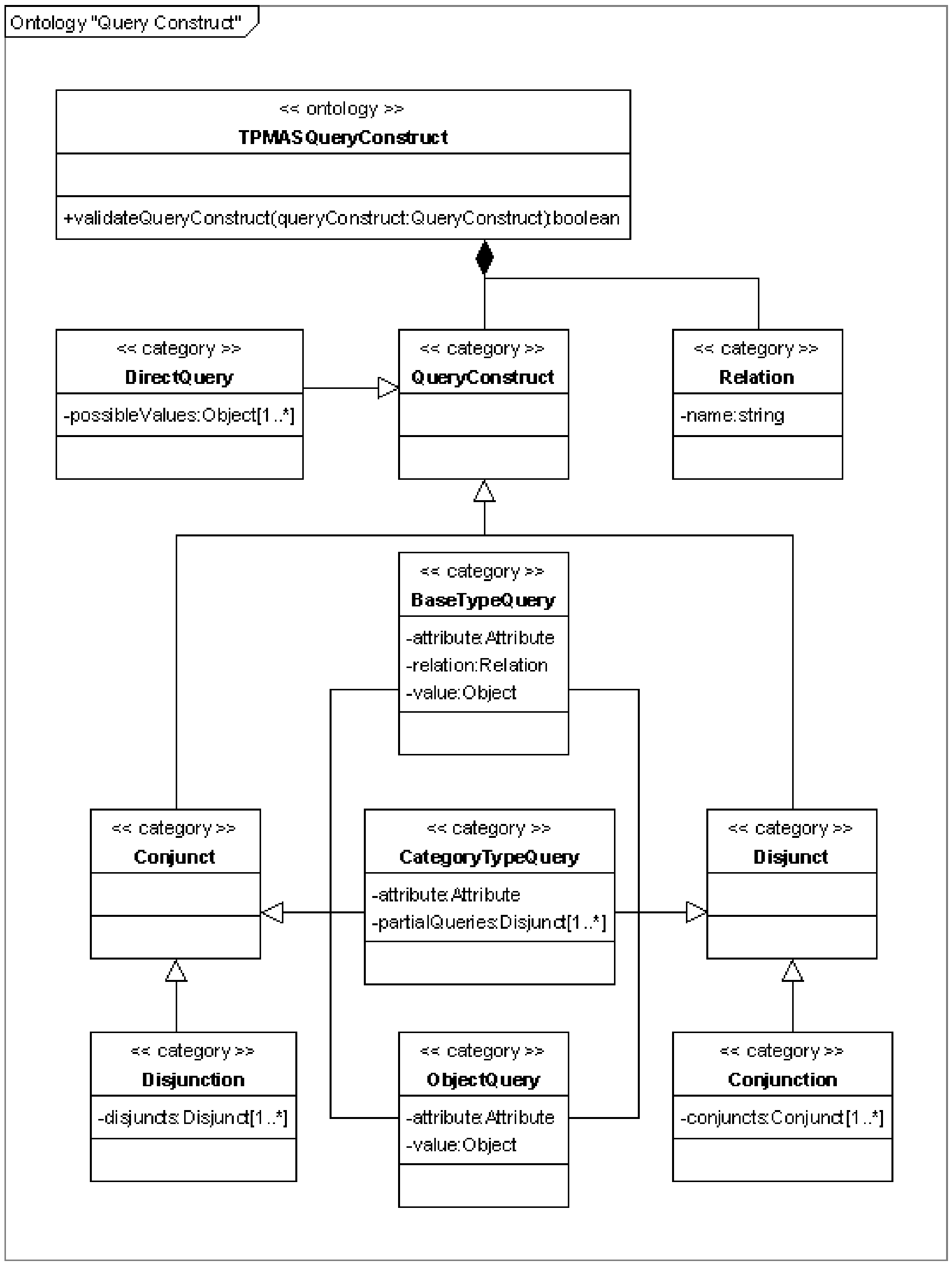


Figure A.4: Diagram for the categories of the ontology "Query Construct".

Table A.16: The interaction CreateContext.

Initiator:	AgentRole
Partner:	TPMASProviderRole
Input:	Identifier of the context to be created
Output:	Context object including information for authorization (if the context could be created), status information
Description:	Interaction for creating a context. As a result, the context is created (if possible).

Table A.17: The interaction TerminateContext.

Initiator:	AgentRole
Partner:	TPMASProviderRole
Input:	Identifier of the context to be terminated, password for full access
Output:	Status information
Description:	Interaction for terminating a context. As a result, the context is terminated (if possible).

Table A.18: The interaction ModifyObjects.

Initiator:	AgentRole
Partner:	TPMASProviderRole
Input:	Identifier of the context to be accessed, password for read/write access, operation to be carried out (store objects, update objects, or remove objects), list of objects
Output:	Status information
Description:	Interaction for accessing a context. As a result, the given operation is carried out (if possible).

Table A.19: The interaction RetrieveObjects.

Initiator:	AgentRole
Partner:	TPMASProviderRole
Input:	Identifier of the context to be accessed, password for read-only access, query construct
Output:	Status information
Description:	Interaction for retrieving objects from a context. As a result, the given query is executed (if possible).

Table A.20: The role schema for the TPMASPROVIDERROLE.

Description:	Provides functionality for transparent persistence in MAS architectures
Protocols: *Activities:*	$\mathsf{CreateContext}_{R,P}$, $\mathsf{TerminateContext}_{R,P}$, $\mathsf{ModifyObjects}_{R,P}$, $\mathsf{RetrieveObjects}_{R,P}$ $\underline{\underline{\mathsf{CreateContext}}}$, $\underline{\underline{\mathsf{TerminateContext}}}$, $\underline{\underline{\mathsf{ModifyObjects}}}$, $\underline{\underline{\mathsf{RetrieveObjects}}}$
Lifeness:	TPMAS_1, TPMAS_2, TPMAS_3, TPMAS_4
$\text{TPMAS}_1 = (\mathsf{CreateContext}_R.\underline{\underline{\mathsf{CreateContext}}}.\mathsf{CreateContext}_P)^\omega$	
$\text{TPMAS}_2 = (\mathsf{TerminateContext}_R.\underline{\underline{\mathsf{TerminateContext}}}.\mathsf{TerminateContext}_P)^\omega$	
$\text{TPMAS}_3 = (\mathsf{ModifyObjects}_R.\underline{\underline{\mathsf{ModifyObjects}}}.\mathsf{ModifyObjects}_P)^\omega$	
$\text{TPMAS}_4 = (\mathsf{RetrieveObjects}_R.\underline{\underline{\mathsf{RetrieveObjects}}}.\mathsf{RetrieveObjects}_P)^\omega$	

A.3 The Recommender Module

This section contains specification related to the Recommender Module described in Chapter 7.

A.3.1 Ontologies

Figure A.5 shows the categories of the ontology "Information Filtering".

A.3.2 Interactions

The interactions of the Information Collection stage are specified in the following tables: Table A.21 describes the interaction UpdateProfile. Table A.22 describes the interaction QueryProfile.

Table A.21: The interaction UpdateProfile.

Initiator:	InterfaceRole
Partner:	ProfileManagerRole
Input:	A list of elements, the identifier of the profile, the operation to be carried out (store objects, update objects, remove objects).
Output:	Status information.
Description:	Interaction for updating a profile. As a result, the profile managed by the ProfileManagerRole is created or updated using the given elements.

Table A.22: The interaction QueryProfile.

Initiator:	InterfaceRole, TFERole
Partner:	ProfileManagerRole
Input:	A query to be applied, the identifier of the profile.
Output:	Elements matching the query (if there are any).
Description:	Interaction for querying a profile.

The interactions of the Information Processing stage are specified in the following tables: Table A.23 describes the interaction ObtainTFE. Table A.24 describes the interaction SetUpdatePolicy. Table A.25 describes the interaction UpdateProfileModel. Table A.26 describes the interaction QueryProfileModel. Table A.27 describes the interaction ModifyProfileModel.

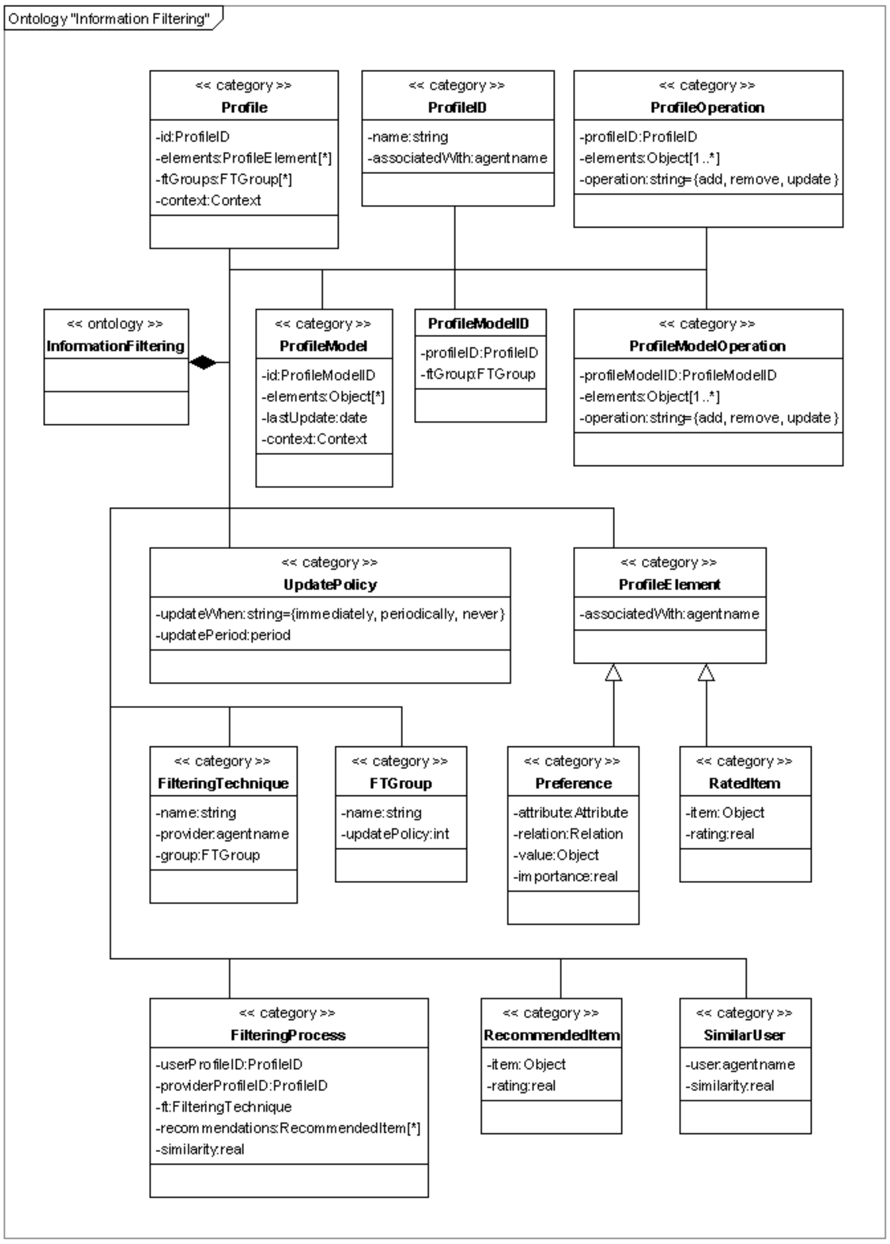

Ontology "Information Filtering"
<< category >> Profile
-id:ProfileID
-elements:ProfileElement[*]
-ftGroups:FTGroup[*]
-context:Context
<< category >> ProfileID
-name:string
-associatedWith:agentname
<< category >> ProfileOperation
-profileID:ProfileID
-elements:Object[1..*]
-operation:string={add, remove, update}
<< ontology >> InformationFiltering
<< category >> ProfileModel
-id:ProfileModelID
-elements:Object[*]
-lastUpdate:date
-context:Context
ProfileModelID
-profileID:ProfileID
-ftGroup:FTGroup
<< category >> ProfileModelOperation
-profileModelID:ProfileModelID
-elements:Object[1..*]
-operation:string={add, remove, update}
<< category >> UpdatePolicy
-updateWhen:string={immediately, periodically, never}
-updatePeriod:period
<< category >> ProfileElement
-associatedWith:agentname
<< category >> FilteringTechnique
-name:string
-provider:agentname
-group:FTGroup
<< category >> FTGroup
-name:string
-updatePolicy:int
<< category >> Preference
-attribute:Attribute
-relation:Relation
-value:Object
-importance:real
<< category >> RatedItem
-item:Object
-rating:real
<< category >> FilteringProcess
-userProfileID:ProfileID
-providerProfileID:ProfileID
-ft:FilteringTechnique
-recommendations:RecommendedItem[*]
-similarity:real
<< category >> RecommendedItem
-item:Object
-rating:real
<< category >> SimilarUser
-user:agentname
-similarity:real

Figure A.5: Diagram for the categories of the ontology "Information Filtering".

Table A.23: The interaction ObtainTFE.

Initiator:	ProfileManagerRole, RelayRole
Partner:	TFEFactoryRole
Input:	The name of a filtering technique, the target platform. Optionally, key sharing data.
Output:	Status information.
Description:	Interaction for obtaining a TFE. As a result, a TFE agent capable of carrying out the given filtering technique is created on the given platform, if possible.

Table A.24: The interaction SetUpdatePolicy.

Initiator:	InterfaceRole
Partner:	ProfileManagerRole
Input:	A group of filtering techniques, an update policy specifying whether to update the profile model periodically, immediately whenever the profile itself changes, or not at all (because the filtering technique is not applied any more).
Output:	Status information.
Description:	Interaction for implementing update policy, which is done as the result.

Table A.25: The interaction UpdateProfileModel.

Initiator:	TFERole
Partner:	ProfileManagerRole
Input:	A list of elements, the identifier of the profile model, the operation to be carried out (store objects, update objects, remove objects).
Output:	Status information.
Description:	Interaction for updating a profile model. As a result, the profile model managed by the role is created or updated using the given elements.

Table A.26: The interaction QueryProfileModel.

Initiator:	TFERole, RelayRole
Partner:	ProfileManagerRole, RelayRole
Input:	A query to be applied, the identifier of the profile model.Alternatively, request for a key to decrypt previously received query results
Output:	Elements matching the query (if there are any). The elements may be encrypted. Alternatively, requested decryption key.
Description:	Interaction for querying a profile model.

Table A.27: The interaction ModifyProfileModel.

Initiator:	ProfileManagerRole
Partner:	TFERole
Input:	A list of elements, the identifier of the profile model, the operation to be carried out (store objects, update objects, remove objects).
Output:	Status information.
Description:	Interaction for modifying a profile model. As a result, the profile model is created or updated using the given elements.

The interactions of the Information Processing stage are specified in the following tables: Table A.28 describes the interaction GetResultsInternally. Table A.29 describes the interaction GetResults. Table A.30 describes the interaction GetResultsAsSupplier. Table A.31 describes the interaction GetResultsAsUser. Table A.32 describes the interaction ExchangeResults. Table A.33 describes the interaction ObtainRelay. Table A.34 describes the interaction ShareKeys.

A.3.3 Role Model

Table A.35 describes the role schema for the InterfaceRole. Table A.36 describes the role schema for the ProfileManagerRole. Table A.37 describes the role schema for the TFEFactoryRole. Table A.38 describes the role schema for the TFERole.

Table A.28: The interaction GetResultsInternally.

Initiator:	INTERFACEROLE^{User}
Partner:	PROFILEMANAGERROLE^{User}
Input:	The name of the filtering technique, the address of the agent realizing the PROFILEMANAGERROLE^{Supplier}, (optionally) an item.
Output:	A list of recommendations or a prediction for the given item, status information.
Description:	Interaction for obtaining personalized information.

Table A.29: The interaction GetResults.

Initiator:	PROFILEMANAGERROLE^{User}
Partner:	PROFILEMANAGERROLE^{Supplier}, RELAYROLE^{Supplier}
Input:	The name of the filtering technique, (optionally) an item.
Output:	A list of recommendations or a prediction for the given item, status information
Description:	Interaction for obtaining personalized information.

Table A.30: The interaction GetResultsAsSupplier.

Initiator:	PROFILEMANAGERROLE^{Supplier}, RELAYROLE^{Supplier}
Partner:	RELAYROLE^{User}
Input:	The name of the filtering technique, (optionally) an item.
Output:	A list of recommendations or a prediction for the given item, status information.
Description:	Interaction for obtaining personalized information.

Table A.31: The interaction GetResultsAsUser.

Initiator:	RelayRole^{User}
Partner:	TFERole
Input:	The name of the filtering technique, (optionally) an item.
Output:	A list of recommendations or a prediction for the given item, status information.
Description:	Interaction for obtaining personalized information.

Table A.32: The interaction ExchangeResults.

Initiator:	ProfileManagerRole^{User}
Partner:	ProfileManagerRole^{Supplier}
Input:	A list of recommendations or a prediction for a given item.
Output:	Status information.
Description:	Interaction for exchanging personalized information.

Table A.33: The interaction ObtainRelay.

Initiator:	ProfileManagerRole^{User}, ProfileManagerRole^{Supplier}
Partner:	ProfileManagerRole^{Supplier}, ProfileManagerRole^{User}
Input:	The target platform.
Output:	Status information.
Description:	Interaction for obtaining a relay. As a result, an agent implementing the respective RelayRole is created on the given platform, if possible.

Table A.34: The interaction ShareKeys.

Initiator:	ProfileManagerRole^{User}, ProfileManagerRole^{Supplier}, TFE-FactoryRole
Partner:	ProfileManagerRole^{Supplier}, RelayRole^{User}, RelayRole^{Supplier}, TFE-Role
Input:	Keys to be shared.
Output:	Status information.
Description:	Interaction for sharing keys to be used in encryption schemes.

Table A.35: The role schema for the InterfaceRole.

Description:	Provides functionality for user interaction (via a GUI or an API).
Protocols: *Activities:*	SetUpdatePolicy$_I$, UpdateProfile$_I$, GetResultsInternally$_I$ $\underline{\underline{\text{ReceiveCommand}}}$
Lifeness:	I_1
$\text{I}_1 = (\underline{\underline{\text{ReceiveCommand}}}.(\text{UpdateProfile}_I \parallel \text{SetUpdatePolicy}_I \parallel \text{GetResultsInternally}_I))^{\omega}$	

Table A.36: The role schema for the PROFILEMANAGERROLE.

Description:	Provides functionality for accessing and modifying profiles and profile models.
Protocols: *Activities:*	UpdateProfile$_{R,P}$, SetUpdatePolicy$_{R,P}$, QueryProfileModel$_{R,P}$, UpdateProfileModel$_{R,P}$, GetResultsInternally$_{R,P}$, GetResults$_{R,P,I}$, ExchangeResults$_{R,P,I}$, ObtainRelay$_{R,P,I}$, ShareKeys$_{R,P,I}$, CreateContext$_{I}$, ModifyObjects$_{I}$, ObtainTFE$_{I}$, RetrieveObjects$_{I}$, ModifyProfileModel$_{I}$, GetResultsAsSupplier$_{I}$, CreateAgent$_{I}$, RestrictCommunication$_{I}$, RevokeControlAndTerminate$_{I}$ $\underline{\underline{\text{CheckProfileInformation}}}$, $\underline{\underline{\text{UpdateProfileInformation}}}$, $\underline{\underline{\text{SetUpdatePolicy}}}$, $\underline{\underline{\text{CheckUpdatePolicy}}}$, $\underline{\underline{\text{UpdateStatistics}}}$, $\underline{\underline{\text{StoreKeys}}}$
Lifeness:	PMR$_1$, PMR$_2$, PMR$_3$, PMR$_4$, PMR$_5$, PMR$_6$, PMR$_7$, PMR$_8$, PMR$_9$, PMR$_1$0
PMR$_1$ = (UpdateProfile$_R$.$\underline{\underline{\text{CheckProfileInformation}}}$.CreateContext$_I$*. $\underline{\underline{\text{UpdateProfileInformation}}}$*.ModifyObjects$_I$.PMR$_3$.UpdateProfile$_P$)$^\omega$	
PMR$_2$ = ($\underline{\underline{\text{CheckUpdatePolicy}}}$.ObtainTFE$_I$.*RetrieveObjects$_I$*. ModifyProfileModel$_I$*.RevokeControlAndTerminate$_I$*)	
PMR$_3$ = (QueryProfileModel$_R$.$\underline{\underline{\text{CheckProfileInformation}}}$. RestrictCommunication$_I$*.RetrieveObjects$_I$*.QueryProfileModel$_P$)$^\omega$	
PMR$_4$ = (UpdateProfileModel$_R$.$\underline{\underline{\text{CheckProfileInformation}}}$. ModifyObjects$_I$*.UpdateProfileModel$_P$)$^\omega$	
PMR$_5$ = (SetUpdatePolicy$_R$.$\underline{\underline{\text{SetUpdatePolicy}}}$.SetUpdatePolicy$_P$)$^\omega$	
PMR$_6$ = (GetResultsInternally$_R$.ObtainRelay$_I$*. GetResults$_I$.RevokeControlAndTerminate$_I$*.ExchangeResults$_I$*. GetResultsInternally$_P$)$^\omega$	
PMR$_7$ = (GetResults$_R$.ObtainRelay$_I$.ObtainTFE$_I$.* GetResultsAsSupplier$_I$.RevokeControlAndTerminate$_I$.$\underline{\underline{\text{UpdateStatistics}}}$. GetResults$_P$)$^\omega$	
PMR$_8$ = (ExchangeResults$_R$.$\underline{\underline{\text{UpdateStatistics}}}$.ExchangeResults$_P$)$^\omega$	
PMR$_9$ = (ObtainRelay$_R$.CreateAgent$_I$.ShareKeys$_I$.ObtainRelay$_P$)$^\omega$	
PMR$_1$0 = (ShareKeys$_R$.$\underline{\underline{\text{StoreKeys}}}$.ShareKeys$_P$)$^\omega$	

Table A.37: The role schema for the TFEFactoryRole.

Description:	Provides agents realizing the TFERole
Protocols: *Activities:*	ObtainTFE$_{R,P}$, CreateAgent$_I$, ShareKeys$_I$
Lifeness:	TFEF$_1$
TFEF$_1$ = (ObtainTFE$_R$.CreateAgent$_I$.ShareKeys$_I$.ShareKeys$_I$. ObtainTFE$_P$)$^\omega$	

Table A.38: The role schema for the TFERole.

Description:	Provides a filtering algorithm to be applied in the Information Processing stage and the Information Filtering stage.
Protocols: *Activities:*	ModifyProfileModel$_{R,P}$, GetResultsAsUser$_{R,P}$, ShareKeys$_{R,P}$, QueryProfileModel$_I$, UpdateProfileModel$_I$ $\underline{\underline{\text{StoreKeys}}}$
Lifeness:	TFE$_1$, TFE$_2$, TFE$_3$
TFE$_1$ = (ModifyProfileModel$_R$.QueryProfileModel$_I$. UpdateProfileModel$_I$.ModifyProfileModel$_P$)$^\omega$	
TFE$_2$ = (GetResultsAsUser$_R$.QueryProfileModel$_I$.QueryProfileModel$_I$. GetResultsAsUser$_P$)$^\omega$	
TFE$_3$ = (ShareKeys$_R$.$\underline{\underline{\text{StoreKeys}}}$.ShareKeys$_P$)$^\omega$	

Table A.39: The role schema for the RelayRole.

Description:	Relays interactions during the Information Filtering stage.
Protocols: *Activities:*	GetResults$_{R,P}$, GetResultsAsSupplier$_{R,P,I}$, QueryProfileModel$_{R,P}$, ShareKeys$_{R,P}$, GetResultsAsUser$_I$, ObtainTFE$_I$, RestrictCommunication$_I$, RevokeControlAndTerminate$_I$ $\underline{\underline{\text{StoreKeys}}}$
Lifeness:	R$_1$, R$_2$, R$_3$
R$_1$ = (GetResults$_R$.ObtainRelay$_I$.ObtainTFE$_I$.GetResultsAsSupplier$_I$. RevokeControlAndTerminate$_I$.GetResults$_P$)$^\omega$	
R$_2$ = (GetResultsAsSupplier$_R$.GetResultsAsUser$_I$. RevokeControlAndTerminate$_I$.GetResultsAsSupplier$_P$)$^\omega$	
R$_3$ = (QueryProfileModel$_R$.RestrictCommunication$_I$*. QueryProfileModel$_I$.QueryProfileModel$_P$)$^\omega$	
R$_4$ = (ShareKeys$_R$.$\underline{\underline{\text{StoreKeys}}}$.ShareKeys$_P$)$^\omega$	

A.4 The Matchmaker Module

A.4.1 Ontologies

Figure A.6 shows the categories of the ontology "Distributed Information Filtering".

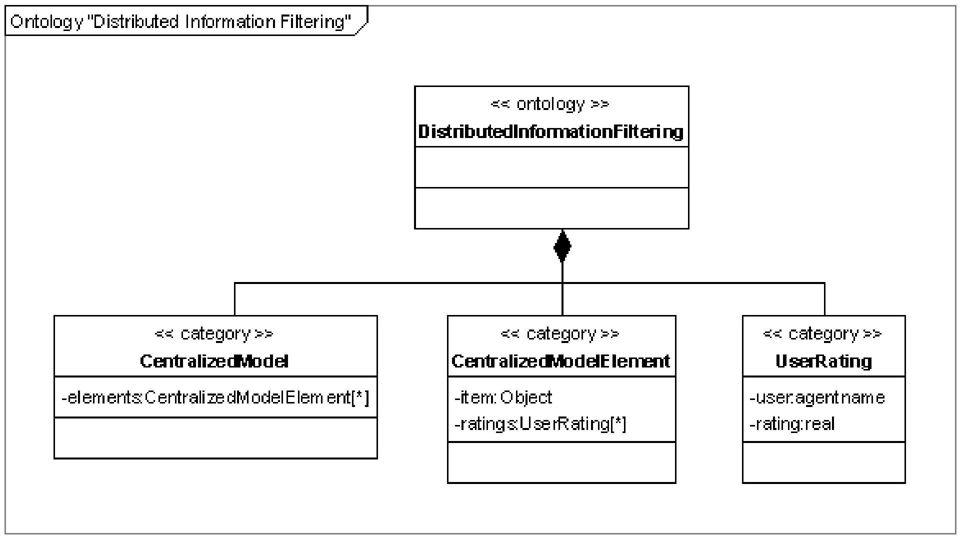


Figure A.6: Diagram for the categories of the ontology "Distributed Information Filtering".

A.4.2 Interactions

The interactions of the Matchmaker Module are specified in Table A.40, which describes the interaction AnnounceProfileElement.

Table A.40: The interaction "AnnounceProfileElement".

Initiator:	PROFILEMANAGERROLE^{User}
Partner:	CENTRALIZEDMODELMANAGERROLE
Input:	User profile element.
Output:	References to other user entities which have announced the same element.
Description:	Interaction for announcing a profile element.

A.4.3 Role Model

Table A.41 describes the role schema for the CentralizedModelManagerRole.

Table A.41: The role schema for the CentralizedModelManagerRole.

Description:	Manages relations of pseudonymized users and profile elements.
Protocols: *Activities:*	AnnounceProfileElement$_{R,P}$, ModifyProfileModel$_{I}$, $\underline{\underline{\text{DetermineOtherUsers}}}$
Lifeness:	CMM$_1$
CMM$_1$ = (AnnounceProfileElement$_R$.ModifyProfileModel$_I$. $\underline{\underline{\text{DetermineOtherUsers}}}$.AnnounceProfileElement$_P$)$^{\omega}$	

Appendix B

Basic Infrastructure: Examples

This appendix illustrates the interactions for controlling communication specified in Section 5.2.2.1. We give examples for interactions in the order they are specified, starting with the four basic interactions of the use case "Restrict Communication", as shown in Figure 5.1. The examples build on each other, i.e. the global state at the end of one example is the global state at the start of the next example. Figure B.1 gives an overview of the controlled platforms after each example.

B.1 Basic Interactions

The agent A_1 on platform P_1 intends to control the agents on platforms P_2 and P_3, neither of which is controlled yet. He uses the interaction RestrictCommunication offered by the supervisor S_1 on its local platform P_1. The supervisor S_1 creates two rules, one for each target platform, and uses the service CheckRule offered by each supervisor on the respective target platform. These supervisors in turn ask all agents on the respective target platforms to consent, via the interaction AcquireConsent. In our example, all agent actually consent, and subsequently the rules are activated, via the service ActivateRule, and the effective rules are determined. Service usage is ended by providing status information, and the supervisor S_1 updates itself by adding the two foreign rules. Finally, all rules are in place as shown in Table B.1.

In the second example, the agent A_1 decides to restrict communication further and intends to control the platform P_2. This effectively leads to agents on both platforms P_2 and P_3 being able only to communicate with A_1 itself, as shown in Table B.2. While the supervisor S_3 still allows communication with P_2, this is blocked by the supervisor S_2 itself.

Table B.1: Example 1: Restrict Communication (I)

Action: A_1 controls P_2, P_3	
Rules of S_1:	
foreign	A_1 controls P_2, exceptions: P_3, A_1
	A_1 controls P_3, exceptions: P_2, A_1
activated	-
effective	-
Rules of S_2:	
foreign	-
activated	A_1 controls P_2, exceptions: P_3, A_1
effective	block communication with exceptions: P_3, A_1
	block attempts to control with exceptions: P_3
Rules of S_3:	
foreign	-
activated	A_1 controls P_3, exceptions: P_2, A_1
effective	block communication with exceptions: P_2, A_1
	block attempts to control with exceptions: P_2

Table B.2: Example 2: Restrict Communication (II)

Action: A_1 controls P_2	
Rules of S_1:	
foreign	A_1 controls P_2, exceptions: P_3, A_1
	A_1 controls P_3, exceptions: P_2, A_1
	A_1 controls P_2, exceptions: A_1
activated	-
effective	-
Rules of S_2:	
foreign	-
activated	A_1 controls P_2, exceptions: P_3, A_1
	A_1 controls P_2, exceptions: A_1
effective	block communication with exceptions: A_1
	block all attempts to control
Rules of S_3 remain unchanged.	

Table B.3: Example 3: Restrict Communication (III)

Action: A_2 controls P_3, P_5	
Rules of S_1 remain unchanged.	
Rules of S_2 remain unchanged.	
Rules of S_3:	
foreign	-
activated	A_1 controls P_3, exceptions: P_2, A_1
	A_2 controls P_3, exceptions: P_5, A_2 (not evaluated for first part of the effective rule)
effective	block communication with exceptions: P_2, A_1
	block all attempts to control
Rules of S_4:	
foreign	A_2 controls P_3, exceptions: P_5, A_2
	A_2 controls P_5, exceptions: P_3, A_2
activated	-
effective	-
Rules of S_5:	
foreign	-
activated	A_2 controls P_5, exceptions: P_3, A_2
effective	block communication with exceptions: P_3, A_2
	block attempts to control with exceptions: P_3

In the third example, the agent A_2 on platform P_4 intends to control the platforms P_3 and P_5. Because P_3 is already controlled by a different agent, the respective rule is not evaluated when determining the effective rule for this platform. Irrespective of this, the platform P_5 is controlled in the usual way. The outcome of this step is shown in Table B.3.

B.2 Revoking Control

Continuing the overall scenario with a fourth example, the agent A_1 intends to revoke control of the platforms P_2 and P_3. Its control of the platform P_2 alone is not affected by this revocation, because it is specified via a different rule. The activated rules related to the group of platforms P_2 and P_3 are removed, and the effective rules are updated. While the platform P_3 is no longer controlled by A_1, it is still controlled by the agent A_2, who is now the first controller and therefore may actually communicate with the platform. The outcome of this step is shown in Table B.4.

Table B.4: Example 4: Revoke Control

Action: A_1 revokes control of P_2, P_3	
Rules of S_1:	
foreign	A_1 controls P_2, exceptions: A_1
activated	-
effective	-
Rules of S_2:	
foreign	-
activated	A_1 controls P_2, exceptions: A_1
effective	block communication with exceptions: A_1 block all attempts to control
Rules of S_3:	
foreign	-
activated	A_2 controls P_3, exceptions: P_5, A_2
effective	block communication with exceptions: P_5, A_2 block attempts to control with exceptions: P_5
Rules of S_4 remain unchanged.	
Rules of S_5 remain unchanged.	

This example also illustrates why the second part of the effective rule is used when deciding on an attempt to control a platform: If an agent would be allowed to control platforms excepted in the first part of the effective rule, an agent on platform P_3 would have been allowed to control platform P_2 after the third example, because the activated rule related to the controller A_2 is not evaluated when determining the first part of the effective rule. This would lead to a conflict when revoking control of the platform A_3, as in this example, because in that case an agent on P_3 would control a platform outside the group of controlled platforms, which is not possible.

B.3 Cascading Control

In the fifth example, A_2 on P_4 intends to control the platform P_1. Because an agent on P_1 controls another platform P_2, control is extended to this platform, through cascading control. The outcome of this step is shown in Table B.5. Note that the resulting rules are the same as in a scenario where the agent A_2 intends to control the platforms P_1 and P_2 directly, because there is no need for additional types of roles for cascading control scenarios. Consequently, there is no equivalent to the cascading control scenario when rules are revoked. If, for example, the agent A_1 revokes control of the platform P_2, the agent A_2 still controls the platform, which makes sense because sensitive information may have been passed between agents on platform P_1 (which the agent A_2 originally intended to control) and P_2 (which the agent A_2 did not explicitly intend to control).

Table B.5: Example 5: Cascading Control

Action: A_2 controls P_1, P_2 (P_2 through cascading control)	
Rules of S_1:	
foreign	A_1 controls P_2, exceptions: A_1
activated	A_2 controls P_1, exceptions: P_2, A_2
effective	block communication with exceptions: P_2, A_2
	block attempts to control with exceptions: P_2
Rules of S_2:	
foreign	-
activated	A_1 controls P_2, exceptions: A_1
	A_2 controls P_2, exceptions: P_1, A_2 (not evaluated for first part of the effective rule)
effective	block communication with exceptions: A_1
	block all attempts to control
Rules of S_3 remain unchanged.	
Rules of S_4:	
foreign	A_2 controls P_3, exceptions: P_5, A_2
	A_2 controls P_5, exceptions: P_3, A_2
	A_2 controls P_1, exceptions: P_2, A_2
	A_2 controls P_2, exceptions: P_1, A_2
activated	-
effective	-
Rules of S_5 remain unchanged.	

B.4 Additional Management Functionality

We conclude with an example illustrating the usage of the additional management services. As an alternate operation to revoking control in the forth example, the agent S_1 instead decides to revoke control and terminate the platforms P_2 and P_3. Because the agent still controls P_2 separately, this platform is not terminated immediately, but an activated rule is added stating that the platform P_2 will be terminated as soon as this separate control is revoked (even if it is then revoked without explicitly terminating the platform). The platform P_3 is controlled by another agent, A_2, but the agent A_1 is the first controller and therefore the platform is terminated immediately. The supervisor S_4 of the platform P_4 (the platform A_2 is located on) is notified and updates its foreign rules accordingly. Additionally, the supervisors of all platforms appearing in the removed foreign roles (in this case, S_5) are notified as well and update their activated and effective rules accordingly. The outcome of this step is shown in Table B.6.

Table B.6: Example 4a: Revoke Control and Terminate

Action: A_1 revokes control of and terminates P_2, P_3	
Rules of S_1:	
foreign	A_1 controls P_2, exceptions: A_1
activated	-
effective	-
Rules of S_2:	
foreign	-
activated	A_1 controls P_2, exceptions: A_1
	A_1 has initiated termination of P_2 (termination is carried out as soon as there are no other activated roles regarding A_1)
effective	block communication with exceptions: A_1
	block all attempts to control
Platform supervised by S_3 has been terminated.	
Rules of S_4:	
foreign	A_2 controls P_3, exceptions: P_5, A_2
	A_2 controls P_5, exceptions: P_3, A_2
	A_2 controls P_1, exceptions: P_2, A_2
	A_2 controls P_2, exceptions: P_1, A_2
activated	-
effective	-
Rules of S_5:	
foreign	-
activated	A_2 controls P_5, exceptions: A_2
effective	block communication with exceptions: A_2
	block all attempts to control

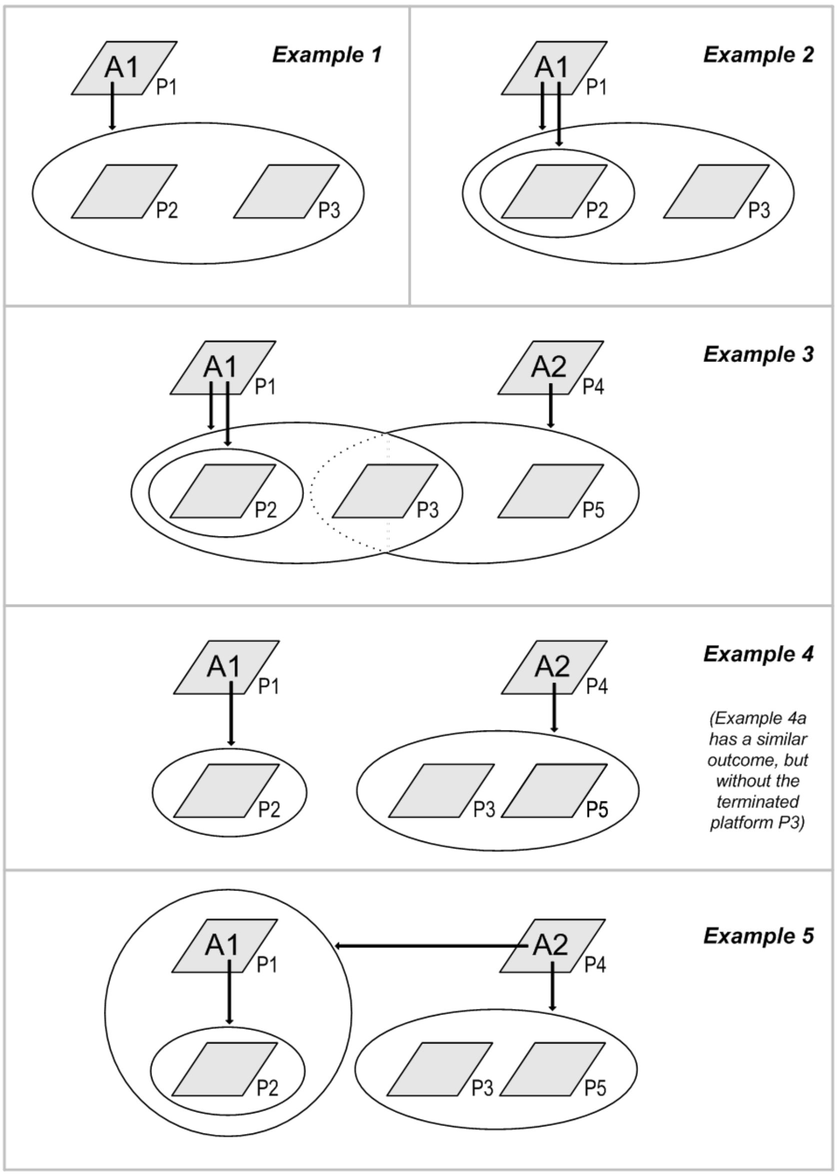


Figure B.1: Controlled platforms after each of the given examples. Platforms are symbolized by parallelograms and labeled Pn, agents are labeled An. Control is indicated by an arrow pointing from the controller to the group of controlled platforms. Communication across solid lines is blocked (except for communication with the controller), dotted lines do not block communication in general, only attempts to control further platforms.

Appendix C

Exemplary Filtering Techniques: Examples

This appendix provides a brief example for hierarchical agglomerative clustering via single-link clustering, as described in Section 9.3.3. Given a set of elements $S = \{A, B, C, D, E, F, G\}$, where each element has two attributes representing its coordinates in a two-dimensional space as shown in the left part of Figure C.1, the squared distances of the elements are computed as shown in Table C.1. The similarity of two elements is defined as their distance. At the beginning of the clustering process, each elements constitutes a single cluster.

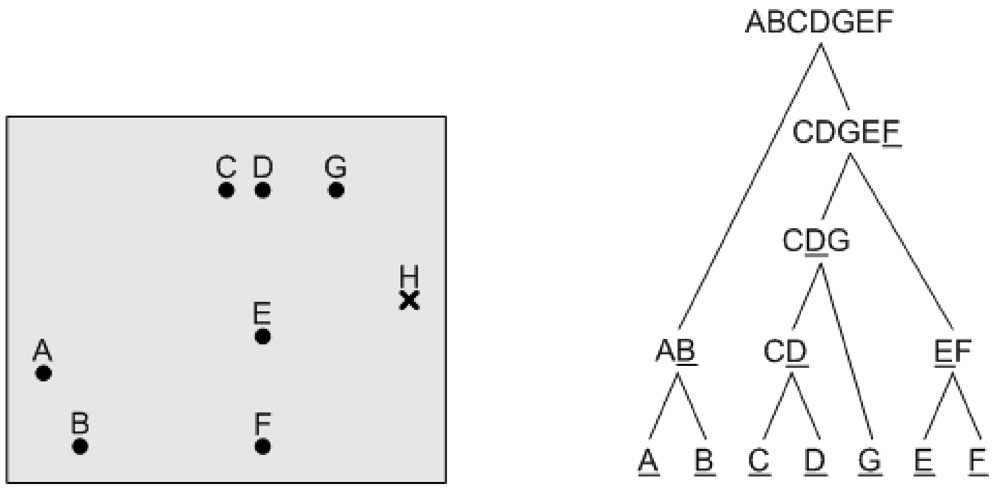


Figure C.1: HAC Example:The positions of the elements and the resulting tree.

Table C.1: HAC Example: The similarity matrix containing the squared distances of the elements.

	A	B	C	D	E	F	G
A	-	5	50	61	37	40	89
B	5	-	65	74	34	25	98
C	50	65	-	1	17	50	9
D	61	74	1	-	16	49	4
E	37	34	17	16	-	9	20
F	40	25	50	49	9	-	53
G	89	98	9	4	20	53	-

In each step of the clustering process, the two clusters with the most similar elements are merged and the similarity matrix is updated, as shown in Table C.2, Table C.3, Table C.4, Table C.5, and Table C.6.

Because cluster representatives are defined as the most similar items between two agglomerated clusters, the cluster representative cannot be determined for clusters that are not yet agglomerated, unless they contain only a single element. The cluster representative candidates are given in parentheses. As an example, in Step 1 the cluster $[CD]$ is created, because the elements C and D have the smallest distance and are therefore most similar. The cluster representative of $[CD]$ cannot be determined yet, but, as indicated in the respective row of Table C.2, it is element C if the cluster is ultimately agglomerated with a cluster containing A or B, and element D otherwise.

Table C.2: HAC Example: The similarity matrix after step 1. C and D are agglomerated into $[CD]$.

	A	B	[CD]	E	F	G
A	-	5	50	37	40	89
B	5	-	65	34	25	98
[CD]	50 (C)	65 (C)	-	16 (D)	49 (D)	4 (D)
E	37	34	16	-	9	20
F	40	25	49	9	-	53
G	89	98	4	20	53	-

The final Step 6 agglomerates all elements into a single cluster. It has to be actually carried out in order to determine the cluster representatives of the final two sub-clusters. Once the tree has been built completely, as shown in the right part of Figure C.1, new elements may be classified by traversing the tree via iteratively selecting the cluster whose representative

Table C.3: HAC Example: The similarity matrix after step 2. $[CD]$ and G are agglomerated into $[[CD]G]$.

	A	B	[[CD]G]	E	F
A	-	5	50	37	40
B	5	-	65	34	25
[[CD]G]	50 (C)	65 (C)	-	16 (D)	49 (D)
E	37	34	16	-	9
F	40	25	49	9	-

Table C.4: HAC Example: The similarity matrix after step 3. A and B are agglomerated into $[AB]$.

	[AB]	[[CD]G]	E	F
[AB]	-	50 (A)	34 (B)	25 (B)
[[CD]G]	50 (C)	-	16 (D)	49 (D)
E	34	16	-	9
F	25	49	9	-

Table C.5: HAC Example: The similarity matrix after step 4. E and F are agglomerated into $[EF]$.

	[AB]	[[CD]G]	[EF]
[AB]	-	50 (A)	25 (B)
[[CD]G]	50 (C)	-	16 (D)
[EF]	25 (F)	16 (E)	-

Table C.6: HAC Example: The similarity matrix after step 1. $[[CD]G]$ and $[EF]$ are agglomerated into $[[[CD]G][EF]]$.

	[AB]	[[[CD]G][EF]]
[AB]	-	25 (B)
[[[CD]G][EF]]	25 (F)	-

is most similar to the new element. As an example the element H shown in the left part of Figure C.1 with squared distances as shown in Table C.7 is classified as follows:

- Choose cluster $[[[CD]G][EF]]$, because H is more similar to cluster representative F than to B.
- Choose cluster $[EF]$, because H is more similar to cluster representative E than to D.
- Choose cluster E, because H is more similar to cluster representative E than to F.

Table C.7: HAC Example: The similarity vector of the new element H.

	A	B	C	D	E	F	G
H	104	97	34	25	17	32	13

In some cases, an element is misclassified as belonging to a cluster c although it is actually more similar to an element of a different cluster c'. In our example, this occurs because H is actually closer to G. In the context of Recommender System functionality, we expect these errors to be generally uncritical.

We conclude the example by discussing privacy aspects. If the provider of the data receives the result E, he does not obtain any additional information regarding H by also receiving the clusters containing E because they constitute information that is known by the provider anyway as long as the provider is able to access the model. Therefore, E may be obtained without giving away H, and without having to retrieve all data, by requesting cluster representatives iteratively and determining similarities independent of the provider of the data.

Appendix D

List of Acronyms

Acronoyms marked with an asterisk are used in this work only. All other acronyms are commonly used.

ACC Agent Communication Channel

ACL Agent Communication Language

AES Advanced Encryption Standard

AOSE Agent-Oriented Software Engineering

AUML Agent Unified Modeling Language

API Application Programming Interface

BIOS Basic Input/Output System

CA Certificate Authority

CRLs Certificate Revocation Lists

CRM Customer Relationship Management

DRM Digital Rights Management

EAL3 Evaluation Assurance Level 3

EJB Enterprise Java Beans

EU European Union

FIPA Foundation for Intelligent Physical Agents

GUI	Graphical User Interface
HMAC	keyed-Hash Message Authentication Code
IEEE	Institute of Electrical and Electronics Engineers
IF	Information Filtering
IR	Information Retrieval
JAAS	Java Authentication and Authorization
JDBC	Java Database Connectivity
JDO	Java Data Objects
JDOQL	Java Data Objects Query Language
JIAC IV	Java Intelligent Agent Componentware, Version IV
JSR	Java Specification Request
JVM	Java Virtual Machine
KDD	Knowledge Discovery in Databases
MAC	Message Authentication Code
MAS	Multi-Agent System
OECD	Organisation for Economic Co-operation and Development
OOSE	Object-Oriented Software Engineering
P3P	Platform for Privacy Preferences Project
PC	Personal Computer
PET	Privacy-Enhancing Technology
PETs	Privacy-Enhancing Technologies
PIR	Private Information Retrieval
PKI	public key infrastructure
PPDM	Privacy-Preserving Data Mining *
PPIF	Privacy-Preserving Information Filtering *

PPTs	Privacy-Preserving Technologies *
RAIC	Redundant Array of Independent Components
RBAC	Role-Based Access Control
RDBMS	Relational Database Management System
RFID	Radio Frequency Identification
SCL	Service Control List
SOA	Service-Oriented Architecture
SPIR	Symmetrically-Private Information Retrieval
SMF1	Secure Message Forwarding Protocol 1 *
SMF2	Secure Message Forwarding Protocol 2 *
SMPC	Secure Multi-Party Computation
SMS	Short Message Service
SQL	Structured Query Language
SSL	Secure Sockets Layer
SVD	Singular Value Decomposition
TCP/IP	Transmission Control Protocol/Internet Protocol
TFE	Temporary Filter Entity *
TPMAS	Transparent Persistence in Multi-Agent Systems *
XML	Extensible Markup Language

Bibliography

[1] D. Agrawal and C. C. Aggarwal. On the Design and Quantification of Privacy Preserving Data Mining Algorithms. In *Proceedings of the Twentieth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 21–23, 2001, Santa Barbara, California, USA*. ACM, 2001.

[2] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic Databases. In *VLDB 2002, Proceedings of 28th International Conference on Very Large Data Bases, August 20–23, 2002, Hong Kong, China*. Morgan Kaufmann, 2002.

[3] R. Agrawal and R. Srikant. Privacy-Preserving Data Mining. In W. Chen, J. F. Naughton, and P. A. Bernstein, editors, *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16–18, 2000, Dallas, Texas, USA*, pages 439–450. ACM, 2000.

[4] E. Aïmeur, G. Brassard, J. M. Fernandez, and F. S. Mani Onana. Privacy-Preserving Demographic Filtering. In H. Haddad, editor, *Proceedings of the 2006 ACM Symposium on Applied Computing (SAC), Dijon, France, April 23–27, 2006*, pages 872–878. ACM, 2006.

[5] J. Algesheimer, C. Cachin, J. Camenisch, and G. Karjoth. Cryptographic Security for Mobile Code. In *2001 IEEE Symposium on Security and Privacy (S&P 2001), May 14–16, 2001, Oakland, California, USA*, pages 2–11. IEEE Computer Society, 2001.

[6] A. Ambainis. Upper Bound on Communication Complexity of Private Information Retrieval. In P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, editors, *Automata, Languages and Programming, 24th International Colloquium, ICALP'97, Bologna, Italy, July 7–11, 1997, Proceedings*, volume 1256 of *Lecture Notes in Computer Science*, pages 401–407. Springer, 1997.

[7] W. A. Arbaugh, D. J. Farber, and J. M. Smith. A Secure and Reliable Bootstrap Architecture. In *1997 IEEE Symposium on Security and Privacy, May 4–7, 1997, Oakland, CA, USA*, pages 65–71. IEEE Computer Society, 1997.

[8] D. Asonov and J. Freytag. Almost Optimal Private Information Retrieval. In R. Dingledine and P. F. Syverson, editors, *Privacy Enhancing Technologies, Second International Workshop, PET 2002, San Francisco, CA, USA, April 14–15, 2002, Revised Papers*, volume 2482 of *Lecture Notes in Computer Science*, pages 209–223. Springer, 2003.

[9] B. Awerbuch, B. Patt-Shamir, D. Peleg, and M. Tuttle. Improved Recommendation Systems. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, January 23–25, 2005*, pages 1174–1183. SIAM, 2005.

[10] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (Im)possibility of Obfuscating Programs. In J. Kilian, editor, *Advances in Cryptology – CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19–23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2001.

[11] B. Bauer, J. P. Müller, and J. Odell. Agent UML: A Formalism for Specifying Multiagent Software Systems. *International Journal of Software Engineering and Knowledge Engineering*, 11(3):207–230, 2001.

[12] C. Bauer and G. King. *Java Persistence with Hibernate.* Manning Publications, 2006.

[13] M. Bawa, R. Bayardo, Jr., and R. Agrawal. Privacy-Preserving Indexing of Documents on the Network. In J. C. Freytag, P. C. Lockemann, S. Abiteboul, M. J. Carey, P. G. Selinger, and A. Heuer, editors, *VLDB 2003, Proceedings of 29th International Conference on Very Large Data Bases, September 9–12, 2003, Berlin, Germany*, pages 922–933. Morgan Kaufmann, 2003.

[14] J. Benaloh. Dense Probabilistic Encryption. In *Selected Areas in Cryptography '94, SAC'94, Kingston, Ontario, Canada, 1994, Proceedings*, pages 120–128, 1994.

[15] F. Brazier and M. Warnier. Organized Anonymous Agents. In *Proceedings of The Third International Symposium on Information Assurance and Security (IAS'07)*, 2007.

[16] E. Brickell, J. Camenisch, and L. Chen. Direct Anonymous Attestation. In V. Atluri, B. Pfitzmann, and P. D. McDaniel, editors, *Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS 2004, Washingtion, DC, USA, October 25–29, 2004*, pages 132–145. ACM, 2004.

[17] K. Bsufka. Public Key Infrastrukturen in Agentenarchitekturen zur Realisierung dienstbasierter Anwendungen, 2006.

[18] Bundesverfassungsgericht (German Federal Constitutional Court). BVerfGE 65,1 – Volkszählung, 1983.

[19] R. Burke. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[20] M. Burrows, M. Abadi, and R. Needham. A Logic of Authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, 1990.

[21] C. Cachin, S. Micali, and M. Stadler. Computationally Private Information Retrieval with Polylogarithmic Communication. In J. Stern, editor, *Advances in Cryptology – EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2–6, 1999, Proceedings*, volume 1592 of *Lecture Notes in Computer Science*, pages 402–414. Springer, 1999.

[22] J. A. Calandrino and A. C. Weaver. Private Resource Pairing. In G. Danezis and P. Golle, editors, *Privacy Enhancing Technologies, 6th International Workshop, PET 2006, Cambridge, UK, June 28–30, 2006, Revised Selected Papers*, volume 4258 of *Lecture Notes in Computer Science*, pages 258–276. Springer, 2006.

[23] J. Camenisch and A. Mityagin. A Mix-Network with Stronger Security. In G. Danezis and D. Martin, editors, *Privacy Enhancing Technologies, 5th International Workshop, PET 2005, Cavtat, Croatia, May 30–June 1, 2005, Revised Selected Papers*, volume 3856 of *Lecture Notes in Computer Science*, pages 128–146. Springer, 2006.

[24] J. Canny. Collaborative Filtering with Privacy. In *2002 IEEE Symposium on Security and Privacy (S&P 2002), May 12–15, 2002, Berkeley, California, USA*, pages 45–57. IEEE Computer Society, 2002.

[25] J. Canny. Collaborative Filtering with Privacy via Factor Analysis. In *SIGIR 2002: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 11–15, 2002, Tampere, Finland*, pages 238–245. ACM, 2002.

[26] J. M. Carroll and M. B. Rosson. Paradox of the Active User. In J. M. Carroll, editor, *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*, chapter 5, pages 80–111. Bradford Books/MIT Press, 1987.

[27] D. Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[28] D. L. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[29] B. Chor and N. Gilboa. Computationally Private Information Retrieval (Extended Abstract). In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing, May 4–6, 1997, El Paso, Texas, USA*, pages 304–313. ACM, 1997.

[30] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private Information Retrieval. In *36th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1995), October 23–25, 1995, Milwaukee, Wisconsin, USA, Proceedings*, pages 41–50. IEEE Computer Society, 1995.

[31] R. Cissée. An Architecture for Agent-Based Privacy-Preserving Information Filtering. In *Proceedings of the 6th International Workshop on Trust, Privacy, Deception and Fraud in Agent Systems*, 2003.

[32] R. Cissée and S. Albayrak. An Agent-Based Approach for Privacy-Preserving Recommender Systems. In E. H. Durfee, M. Yokoo, M. N. Huhns, and O. Shehory, editors, *6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007), Honolulu, Hawaii, USA, May 14–18, 2007*. IFAAMAS, 2007.

[33] R. Clarke. Meta-Brands: Privacy Marks and Seals. *Privacy Law & Policy Reporter*, 8(5), 5 2001.

[34] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu. Tools for Privacy Preserving Distributed Data Mining. *ACM SIGKDD Explorations*, 4(2):28–34, 2002.

[35] R. Cramer, R. Gennaro, and B. Schoenmakers. A Secure and Optimally Efficient Multi-Authority Election Scheme. In W. Fumy, editor, *Advances in Cryptology – EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11–15, 1997, Proceedings*, volume 1233 of *Lecture Notes in Computer Science*, pages 103–118. Springer, 1997.

[36] G. Danezis, R. Dingledine, D. Hopwood, and N. Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *2003 IEEE Symposium on Security and Privacy (S&P 2003), May 11–14 2003, Berkeley, California, USA*, pages 2–15. IEEE Computer Society, 2003.

[37] L. D'Anna, B. Matt, A. Reisse, T. V. Vleck, S. Schwab, and P. LeBlanc. Self-Protecting Mobile Agents Obfuscation Report. Technical Report 03-015, Network Associates Laboratories, 2003. Final report. June 30, 2003.

[38] M. Deshpande and G. Karypis. Item-Based Top-N Recommendation Algorithms. *ACM Transactions on Information Systems*, 22(1):143–177, 2004.

[39] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium*, 2004.

[40] Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the Protection of Individuals with Regard to the Processing of Personal Data and on the Free Movement of such Data, 1995.

[41] P. Drineas, I. Kerenidis, and P. Raghavan. Competitive Recommendation Systems. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing, May 19–21, 2002, Montral, Qubec, Canada*, pages 82–90. ACM, 2002.

[42] T. El-Gamal. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, 1985.

[43] Electronic Privacy Information Center and Privacy International. *Privacy and Human Rights 2006. An International Survey of Privacy Laws and Developments.* Electronic Privacy Information Center, 2006.

[44] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy Preserving Mining of Association Rules. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23–26, 2002, Edmonton, Alberta, Canada*, pages 217–228. ACM, 2002.

[45] L. Foner. *Political Artifacts and Personal Privacy: The Yenta Multi-Agent Distributed Matchmaking System.* PhD thesis, Massachusetts Institute of Technology, 1999.

[46] Foundation for Intelligent Physical Agents. FIPA Abstract Architecture Specification, Version L, 2002.

[47] Foundation for Intelligent Physical Agents. FIPA ACL Message Structure Specification, Version G, 2002.

[48] Foundation for Intelligent Physical Agents. FIPA Agent Message Transport Service Specification, Version F, 2002.

[49] Foundation for Intelligent Physical Agents. FIPA Agent Management Specification, Version K, 2004.

[50] S. Fricke, K. Bsufka, J. Keiser, T. Schmidt, R. Sesseler, and S. Albayrak. Agent-based Telematic Services and Telecom Applications. *Communications of the ACM*, 44(4), April 2001.

[51] T. Garfinkel, M. Rosenblum, and D. Boneh. Flexible OS Support and Applications for Trusted Computing. In M. B. Jones, editor, *Proceedings of HotOS'03: 9th Workshop on Hot Topics in Operating Systems, May 18–21, 2003, Lihue (Kauai), Hawaii, USA*. USENIX, 2003.

[52] T. Geissler and O. Kroll-Peters. Applying Security Standards to Multi Agent Systems. In *AAMAS Workshop: Safety & Security in Multiagent Systems*, 2004.

[53] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting Data Privacy in Private Information Retrieval Schemes. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing, May 23–26, 1998, Dallas, Texasa, USA*, pages 151–160. ACM, 1998.

[54] I. Goldberg. Privacy-Enhancing Technologies for the Internet, II: Five Years Later. In R. Dingledine and P. F. Syverson, editors, *Privacy Enhancing Technologies, Second International Workshop, PET 2002, San Francisco, CA, USA, April 14–15, 2002, Revised Papers*, volume 2482 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2003.

[55] I. Goldberg, D. Wagner, and E. Brewer. Privacy-Enhancing Technologies for the Internet. In *Proceedings of the 42nd IEEE Computer Society International Conference (COMPCON)*. IEEE Computer Society Press, February 1997.

[56] O. Goldreich, S. Micali, and A. Wigderson. How to Play ANY Mental Game. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, New York, New York, USA*, pages 218–229. ACM, 1987.

[57] D. Goldschlag, M. Reed, and P. Syverson. Onion Routing for Anonymous and Private Internet Connections. *Communications of the ACM (USA)*, 42(2):39–41, 1999.

[58] L. Gong, M. Mueller, H. Prafullchandra, and R. Schemers. Going Beyond the Sandbox: An Overview of the New Security Architecture in the Java Development Kit 1.2. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, December 1997.

[59] N. Good, J. B. Schafer, J. A. Konstan, A. Borchers, B. M. Sarwar, J. L. Herlocker, and J. Riedl. Combining Collaborative Filtering with Personal Agents for Better Recommendations. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence, July 18–22, 1999, Orlando, Florida, USA*, pages 439–446. AAAI Press / The MIT Press, 1999.

[60] V. Haldar, D. Chandra, and M. Franz. Semantic Remote Attestation – Virtual Machine Directed Approach to Trusted Computing. In *Proceedings of the 3rd Virtual Machine Research and Technology Symposium, May 6–7, 2004, San Jose, CA, USA*, pages 29–41. USENIX, 2004. Best Paper Award.

[61] J. Halpern and K. O'Neill. Anonymity and Information Hiding in Multiagent Systems. In *16th IEEE Computer Security Foundations Workshop (CSFW-16 2003), June 30–2 July 2, 2003, Pacific Grove, CA, USA*, pages 75–88. IEEE Computer Society, 2003.

[62] J. Hipp, U. Güntzer, and G. Nakhaeizadeh. Algorithms for Association Rule Mining – A General Survey and Comparison. *ACM SIGKDD Explorations*, 2(1):58–64, July 2000.

[63] R. Ingham. Privacy & Psychology. In J. B. Young, editor, *Privacy*, pages 35–57. John Wiley & Sons, 1978.

[64] W. Jansen. Countermeasures for Mobile Agent Security. *Computer Communications*, 23(17):1667–1676, 2000.

[65] Java Data Objects Expert Group. Java(tm) Data Objects 2.0. JSR 243. JSR-000243, 2006. Status: Final Release; Release: 20 March 2006.

[66] S. Jha, L. Kruger, and P. McDaniel. Privacy Preserving Clustering. In S. De Capitani di Vimercati, P. F. Syverson, and D. Gollmann, editors, *Computer Security – ESORICS 2005, 10th European Symposium on Research in Computer Security, Milan, Italy, September 12–14, 2005, Proceedings*, volume 3679 of *Lecture Notes in Computer Science*, pages 397–417. Springer, 2005.

[67] JSR 220 Expert Group. Enterprise JavaBeans(tm) 3.0. JSR 220. JSR-000220, 2006. Status: Final Release; Release: 11 May 2006.

[68] M. Kantarcioglu. Privacy-Preserving Distributed Mining of Association Rules on Horizontally Partitioned Data. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1026–1037, 2004.

[69] G. Karjoth, M. Schunter, and M. Waidner. The Platform for Enterprise Privacy Practices: Privacy-Enabled Management of Customer Data. In R. Dingledine and P. F. Syverson, editors, *Privacy Enhancing Technologies, Second International Workshop, PET 2002, San Francisco, CA, USA, April 14–15, 2002, Revised Papers*, volume 2482 of *Lecture Notes in Computer Science*, pages 69–84. Springer, 2003.

[70] W. Kim. Personalization: Definition, Status, and Challenges Ahead. *Journal of Object Technology*, 1(1), 2002.

[71] A. Kobsa. Generic User Modeling Systems. *User Modeling and User-Adapted Interaction*, 11(1–2):49–63, 2001.

[72] A. Kobsa. A Component Architecture for Dynamically Managing Privacy Constraints in Personalized Web-Based Systems. In R. Dingledine, editor, *Privacy Enhancing Technologies, Third International Workshop, PET 2003, Dresden, Germany, March 26–28, 2003, Revised Papers*, volume 2760 of *Lecture Notes in Computer Science*, pages 177–188. Springer, 2003.

[73] L. Korba, R. Song, and G. Yee. Anonymous Communication for Mobile Agents. In A. Karmouch, T. Magedanz, and J. Delgado, editors, *Mobile Agents for Telecommunication Applications, 4th International*

Workshop, MATA 2002 Barcelona, Spain, October 23–24, 2002, Proceedings, volume 2521 of *Lecture Notes in Computer Science*, pages 171–181. Springer, 2002.

[74] D. Kuokka and L. Harada. Integrating Information via Matchmaking. *Journal of Intelligent Information Systems*, 6(2–3):261–279, 1996.

[75] E. Kushilevitz and R. Ostrovsky. Replication is Not Needed: Single Database, Computationally-Private Information Retrieval. In *38th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1995), October 19–22, 1997, Miami Beach, Florida, USA, Proceedings*, pages 364–373. IEEE Computer Society, 1997.

[76] C. Lai, L. Gong, L. Koved, A. Nadalin, and R. Schemers. User Authentication and Authorization in the Java Platform. In *15th Annual Computer Security Applications Conference (ACSAC 1999), December 6–10, 1999, Scottsdale, AZ, USA*. IEEE Computer Society, 1999.

[77] S. K. Lam, D. Frankowski, and J. Riedl. Do You Trust Your Recommendations? An Exploration of Security and Privacy Issues in Recommender Systems. In G. Müller, editor, *Emerging Trends in Information and Communication Security, International Conference, ETRICS 2006, Freiburg, Germany, June 6–9, 2006, Proceedings*, volume 3995 of *Lecture Notes in Computer Science*, pages 14–29. Springer, 2006.

[78] Y. Lindell and B. Pinkas. Privacy Preserving Data Mining. In M. Bellare, editor, *Advances in Cryptology – CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20–24, 2000, Proceedings*, volume 1880 of *Lecture Notes in Computer Science*, pages 36–54. Springer, 2000.

[79] H. Link, J. Saia, T. Lane, and R. A. LaViolette. The Impact of Social Networks on Multi-Agent Recommender Systems. In *Proceedings of the Workshop on Cooperative Multi-Agent Learning (ECML/PKDD '05)*, 2005.

[80] P. Lyman and H. R. Varian. How Much Information, 2003. Retrieved May 19, 2009, from http://www.sims.berkeley.edu/how-much-info-2003.

[81] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay – A Secure Two-party Computation System. In *Proceedings of the 13th USENIX Security Symposium*, 2004.

[82] S. M. McNee, J. Riedl, and J. A. Konstan. Being Accurate is Not Enough: How Accuracy Metrics Have Hurt Recommender Systems. In G. M. Olson and R. Jeffries, editors, *Extended Abstracts Proceedings of the 2006 Conference on Human Factors in Computing Systems, CHI 2006, Montréal, Québec, Canada, April 22–27, 2006*, pages 1097–1101. ACM, 2006.

[83] Merriam-Webster Online Dictionary, 2008. Retrieved May 19, 2009, from http://www.merriam-webster.com/dictionary/privacy.

[84] S. Merugu and J. Ghosh. Privacy-Preserving Distributed Clustering Using Generative Models. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003), December 19–22, 2003, Melbourne, Florida, USA*, pages 211–218. IEEE Computer Society, 2003.

[85] B. N. Miller, J. A. Konstan, and J. Riedl. PocketLens: Toward a Personal Recommender System. *ACM Transactions on Information Systems*, 22(3):437–476, 2004.

[86] M. Naor and B. Pinkas. Oblivious Polynomial Evaluation. *SIAM Journal on Computing*, 35(5):1254–1281, 2006.

[87] E. M. Noam. Privacy and Self-Regulation: Markets for Electronic Privacy. In *Privacy and Self-Regulation in the Information Age*. U.S. Department of Commerce, 1997.

[88] Online Ethics Center for Engineering and Science. The Online Ethics Center Glossary, 2008. Retrieved May 26, 2008, from http://www.onlineethics.diamax.com/CMS/cms/glossary.aspx.

[89] Organisation for Economic Co-operation and Development. Guidelines on the Protection of Privacy and Transborder Flows of Personal Data. OECD, Paris, 1980.

[90] The Platform for Privacy Preferences 1.0 (P3P1.0) Specification. W3C Recommendation, 2002.

[91] A. Pfitzmann and M. Köhntopp. Anonymity, Unobservability, and Pseudonymity – A Proposal for Terminology. In H. Federrath, editor, *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 25–26, 2000, Proceedings*, volume 2009 of *Lecture Notes in Computer Science*, pages 1–9. Springer, 2001.

[92] H. Polat and W. Du. Privacy-Preserving Collaborative Filtering Using Randomized Perturbation Techniques. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003), December 19–22, 2003, Melbourne, Florida, USA*, pages 625–628. IEEE Computer Society, 2003.

[93] H. Polat and W. Du. SVD-Based Collaborative Filtering with Privacy. In H. Haddad, L. M. Liebrock, A. Omicini, and R. L. Wainwright, editors, *Proceedings of the 2005 ACM Symposium on Applied Computing (SAC), Santa Fe, New Mexico, USA, March 13–17, 2005*, pages 791–795. ACM, 2005.

[94] K. Rannenberg. Multilateral Security. A Concept and Examples for Balanced Security. In *Proceedings of the New Security Paradigms Workshop 2000, Cork, Ireland, USA, September 19–21, 2000*, pages 151–162. ACM, 2000.

[95] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for Web Transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[96] P. Resnick and H. R. Varian. Recommender Systems (Introduction to Special Section). *Communications of the ACM*, 40(3):56–58, march 1997.

[97] V. Roth. Secure Recording of Itineraries through Co-operating Agents. In S. Demeyer and J. Bosch, editors, *Object-Oriented Technology, ECOOP'98 Workshop Reader, ECOOP'98 Workshops, Demos, and Posters, Brussels, Belgium, July 20–24, 1998, Proceedings*, volume 1543 of *Lecture Notes in Computer Science*, pages 297–298. Springer, 1998.

[98] T. Sander and C. Tschudin. Protecting Mobile Agents Against Malicious Hosts. In G. Vigna, editor, *Mobile Agents and Security*, volume 1419 of *Lecture Notes in Computer Science*, pages 44–60. Springer, 1998.

[99] T. Schmidt. Advanced Security Infrastructure for Multi-Agent-Applications in the Telematic Area. Dissertation Thesis, Technische Universität Berlin, 2002.

[100] R. Sesseler. Eine modulare Architektur für dienstbasierte Interaktionen zwischen Agenten. Dissertation Thesis, Technische Universität Berlin, 2002.

[101] R. Sesseler and S. Albayrak. Service-ware Framework for Developing 3G Mobile Services. In *The Sixteenth International Symposium on Computer and Information Sciences, ICSIS XVI*, November 2003.

[102] G. J. Simmons. The Prisoners' Problem and the Subliminal Channel. In D. Chaum, editor, *Advances in Cryptology – CRYPTO '83, A Workshop on the Theory and Application of Cryptographic Techniques, Santa Barbara, California, USA, August 22–24, 1983, Proceedings*, pages 51–67. Plenum Press, 1984.

[103] G. J. Simmons. The Subliminal Channel and Digital Signatures. In T. Beth, N. Cot, and I. Ingemarsson, editors, *Advances in Cryptology: Proceedings of EUROCRYPT 84, A Workshop on the Theory and Application of of Cryptographic Techniques, Paris, France, April 9–11, 1984, Proceedings*, volume 209 of *Lecture Notes in Computer Science*, pages 364–378. Springer, 1985.

[104] S. W. Smith and D. Safford. Practical Server Privacy with Secure Coprocessors. *IBM Systems Journal*, 40(3):683–695, 2001.

[105] Supreme Court of the United States. Olmstead v. United States, 277 U.S. 438, 1928.

[106] L. Sweeney. *k*-Anonymity: A Model for Protecting Privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

[107] H. Taylor. Most People Are "Privacy Pragmatists" Who, While Concerned about Privacy, Will Sometimes Trade It Off for Other Benefits. The Harris Poll No. 17, March 19, 2003.

[108] M. Teltzrow and A. Kobsa. Impacts of User Privacy Preferences on Personalized Systems: A Comparative Study. In *Designing personalized user experiences in eCommerce*, pages 315–332. Kluwer Academic Publishers, Norwell, MA, USA, 2004.

[109] United States Department of Health, Education and Welfare. Secretary's Advisory Committee on Automated Personal Data Systems, Records, Computers, and the Rights of Citizens VIII, 1973.

[110] J. Vaidya and C. Clifton. Privacy Preserving Association Rule Mining in Vertically Partitioned Data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data*

Mining, July 23–26, 2002, Edmonton, Alberta, Canada, pages 639–644. ACM, 2002.

[111] P. van Oorschot. Revisiting Software Protection. In C. Boyd and W. Mao, editors, *Information Security, 6th International Conference, ISC 2003, Bristol, UK, October 1–3, 2003, Proceedings*, volume 2851 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2003.

[112] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis. State-of-the-art in Privacy Preserving Data Mining. *SIGMOD Record*, 33(1):50–57, 2004.

[113] J. M. Vidal. A Protocol for a Distributed Recommender System. In R. Falcone, K. S. Barber, J. Sabater-Mir, and M. P. Singh, editors, *Trusting Agents for Trusting Electronic Societies, Theory and Applications in HCI and E-Commerce*, volume 3577 of *Lecture Notes in Computer Science*, pages 200–217. Springer, 2005.

[114] G. Vigna. Protecting Mobile Agents through Tracing. In *Proceedings of the 3rd ECOOP Workshop on Mobile Object Systems*, 1997.

[115] S. D. Warren and L. D. Brandeis. The Right to Privacy. *Harvard Law Review*, 4(5), December 1890.

[116] A. F. Westin. *Privacy and Freedom.* Atheneum, 1967.

[117] J. Wohltorf, R. Cissée, and A. Rieger. BerlinTainment: An Agent-Based Context-Aware Entertainment Planning System. *IEEE Communications Magazine*, 43(6), 2005.

[118] M. Wooldridge and N. R. Jennings. Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.

[119] A. Yao. Protocols for Secure Computation. In *23rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 1982), November 3–5, 1982, Chicago, Illinois, USA, Proceedings*, pages 160–164. IEEE Computer Society, 1982.

[120] F. Zambonelli, N. R. Jennings, and M. Wooldridge. Developing Multiagent Systems: The Gaia Methodology. *ACM Transactions on Software Engineering and Methodology*, 12(3):317–370, 2003.

[121] N. Zhang and W. Zhao. Distributed Privacy Preserving Information Sharing. In K. Böhm, C. S. Jensen, L. M. Haas, M. L. Kersten, P.-Å.

Larson, and B. C. Ooi, editors, *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30–September 2, 2005*, pages 889–900. ACM, 2005.

[122] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving Recommendation Lists Through Topic Diversification. In A. Ellis and T. Hagino, editors, *Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10–14, 2005*, pages 22–32. ACM, 2005.

Index