FIDIS

Future of Identity in the Information Society

Title:	"D14.3: Study on the Suitability of Trusted Computing to support Privacy in Business Processes"
Author:	WP14
Editors:	Günter Müller, Sven Wohlgemuth (ALU-FR)
Reviewers:	Martin Meints (ICPP), Jozef Vyskoc (VaF)
Identifier:	D14.3
Type:	[Deliverable]
Version:	1.0
Date:	Tuesday, 06 May 2008
Status:	[Final]
Class:	[Public]
File:	fidis_wp14_d14.3_v1.0.doc

Summary

The European Directives 95/46/EC and 2002/58/EC demand the consent of users for a purpose-based processing of their data. In practice, users give their consent to the privacy statements of service providers, if they want to use personalised services. Since current privacy enhancing technologies focus on the disclosure of personal data and not on their usage, users are not able to verify whether service providers follow their privacy statement. It follows that users have to trust service providers that they enforce the rules of their privacy statement.

The objective of this deliverable is to investigate on Trusted Computing whether it is suitable to realise a trust model where service providers are able to show users that they have enforced the agreed rules. The motive for choosing Trusted Computing is that Trusted Computing provides a tamper-resistant foundation for identifying an information system's configuration and so to identify if specific services, e.g. for monitoring the usage of personal data, are used.

Approaches for using Trusted Computing in order to support the enforcement of privacy policies are presented. This deliverable proposes a modification of the specification by the Trusted Computing Group and a monitor for observing the usage of personal data.

Copyright Notice:

Members of the FIDIS consortium

1. Goethe University Frankfurt	Germany
2. Joint Research Centre (JRC)	Spain
3. Vrije Universiteit Brussel	Belgium
4. Unabhängiges Landeszentrum für Datenschutz	Germany
5. Institut Europeen D'Administration Des Affaires (INSEAD)	France
6. University of Reading	United Kingdom
7. Katholieke Universiteit Leuven	Belgium
8. Tilburg University	Netherlands
9. Karlstads University	Sweden
10. Technische Universität Berlin	Germany
11. Technische Universität Dresden	Germany
12. Albert-Ludwig-University Freiburg	Germany
13. Masarykova universita v Brne	Czech Republic
14. VaF Bratislava	Slovakia
15. London School of Economics and Political Science	United Kingdom
16. Budapest University of Technology and Economics (ISTRI)	Hungary
17. IBM Research GmbH	Switzerland
18. Institut de recherche criminelle de la Gendarmerie Nationale	France
19. Netherlands Forensic Institute	Netherlands
20. Virtual Identity and Privacy Research Center	Switzerland
21. Europäisches Microsoft Innovations Center GmbH	Germany
22. Institute of Communication and Computer Systems (ICCS)	Greece
23. AXSionics AG	Switzerland
24. SIRRIX AG Security Technologies	Germany

Versions

Version	*Date*	*Description (Editor)*
0.1	09.07.2007	Initial release with first sketch of table of contents and section 2, 3.1, 3.4, 4.1, , 5.3.1, 5.3.2, 6.1 and 6.2 added (Sven Wohlgemuth)
0.2	02.08.2007	Sections 3.2 (Richard Cissée), 5.1 (Stefan Köpsell), 5.3.3 (Stefan Köpsell) and 6.3 (Richard Cissée) added
0.3	21.09.2007	Sections 4 and 3.3 added (Rani Husseiki).
0.4	02.11.2007	Modifications of the structure (Sven Wohlgemuth)
0.5	29.11.2007	Version for FIDIS internal review (Sven Wohlgemuth)
1.0	06.05.2008	Final version; revised by comments of FIDIS internal review (Sven Wohlgemuth)

Foreword

FIDIS partners from various disciplines have contributed as authors to this document. The following list names the main contributors for the chapters of this document:

Chapter	*Contributor(s)*
1 (Executive Summary)	Sven Wohlgemuth (ALU-FR)
2 (Introduction)	Sven Wohlgemuth (ALU-FR)
3 (The Need for the controllable Processing of Personal Data)	Richard Cissée (TUB), Rani Husseiki (SIRRIX) and Sven Wohlgemuth (ALU-FR)
4 (An Introduction to Trusted Computing)	Rani Husseiki (SIRRIX) and Sven Wohlgemuth (ALU-FR)
5 (Employment of Trusted Computing Platform)	Richard Cissée (TUB), Stefan Köpsell (TUD) and Sven Wohlgemuth (ALU-FR)
6 (Approaches for Using TCP as a Foundation for Policy-Compliant Data Processing)	Richard Cissée (TUB) and Sven Wohlgemuth (ALU-FR)
7 (Results)	Richard Cissée (TUB) and Sven Wohlgemuth (ALU-FR)

Table of Contents

1 Executive Summary

This study refers (a) to the usage of disclosed personal data according to agreed rules between users and service providers and (b) to the suitability of Trusted Computing for enforcing privacy rules. In business processes for personalised services, personal data is used by service providers, e.g. for storing or delegating it to other service providers. The rules for processing personal data are published by service providers by means of privacy statements. Users give their consent to these privacy statements if they agree to the standard business conditions. However, since privacy enhancing technologies (PET) mainly focus on the access on personal data in terms of their collection, users are neither able to enforce the agreed rules nor to verify whether service providers have followed these rules. Consequently, trust of users in service providers according to the enforcement of these rules is mandatory. In order to extend this trust model so that users need not trust service providers in this context, this study presents approaches for technically observing the behaviour of service providers with respect to the agreed privacy rules.

This study introduces this privacy problem by two use cases: information filtering and personalised services. The first one appears in recommender systems and second one in customer relationship management, e.g. loyalty programmes. Both describe a delegation of collected personal data from one service provider to another. In order to enforce rules on the delegation of personal data, this study presents a monitor in combination with a technique of information flow analysis in order to identify delegation of personal data contrary to the agreed rules. In order to verify the information flow by the user, a service provider gives the user a proof that it uses such a monitor and presents him the transcript of the observations. The approach presented in this study makes use of Trusted Computing in reverse in comparison to its common use: Trusted Computing is used in the information system of the service provider and not in the system of the user. The investigation in Trusted Computing discusses its shortcomings for this kind of deployment and presents a solution for the presented “time problem”.

The conclusion that is drawn from this investigation is that delegation and storing activities of a service provider concerning personal data can be observed by the presented monitor in combination with the use of Trusted Computing. For example, medical service providers which make use of the electronic patient record can deploy this approach to show their trustworthiness to their users. Users do not need to trust them anymore that they follow the agreed privacy rules concerning the storage and delegation of personal data.

However, most of the technologies related to Trusted Computing are not widely used or even available today. The specification for usage of Trusted Computing on servers as published by the Trusted Computing group seems to be very preliminary. It does not address problems related to Trusted Computing and modern server side technologies like virtualisation and partitioning. The last revision (0.8) is from March 23rd, 2005. No visible changes seem to happen since then. Moreover many of the proposed security and privacy solutions are based on very clear and static overall processing environments. In practice a dynamic environment (e.g. new version of or patches for the operating system and the business software etc.) will be used. At the moment, it is not clear if and how the integrity measurements (including remote attestation) should work in a dynamic environment.

2 Privacy: A Matter of Trust?

The technological development enables a provision of personalised services via the Internet, e.g. a personalised offer for a health insurance. The collection and use of personal data is therefore inevitable. In spite of the advantages of personalised services for their users, the use of such services bring along privacy threats.

2.1 Trust in Enforcement of Privacy Statements

The European Directives 95/46/EC and 2002/58/EC demand the consent of users for a purpose-based processing of personal data according to informational self-determination. In practice, service providers publish the rules of their collection and usage of personal data by their privacy statement. Users agree to these rules if they want to use personalised services and thereby give their consent to the standard business conditions. An example for this practice are loyalty programmes (Müller and Wohlgemuth, 2007).

Existing privacy enhancing technologies (PET) mainly follow the principle of data economy and do not focus on the enforcement of rules which correspond to disclosed data (Sackmann, Strüker and Accorsi, 2006). Identity management so far focuses in general on the disclosure and with the extension of delegation of rights (Wohlgemuth and Müller, 2006) also on the delegation of personal data where users act with pseudonyms. P3P (Cranor, Langheinrich, Marchiori, Marshall and Reagle, 2002) and EPAL (Ashley, Hada, Karjoth, Powers and Schunter, 2003) formalize privacy statements. But P3P does not consider rules (obligations) on the usage of personal data. EPAL supports obligations but does not offer a technical mechanism enabling users to control their enforcement. Consequently, users are not aware of rule violations. Disclosed personal data may be misused by service providers, e.g. used for other purposes or delegated to other services without the consent of the user.

This current, one-sided trust model is shown in Figure 1. The user discloses personal data *d* to a service provider which uses and delegates it in combination with user's data *d'* to other service provider. Users have to trust that these service providers follow the agreed rules for the processing of personal data *d* and *d'*.

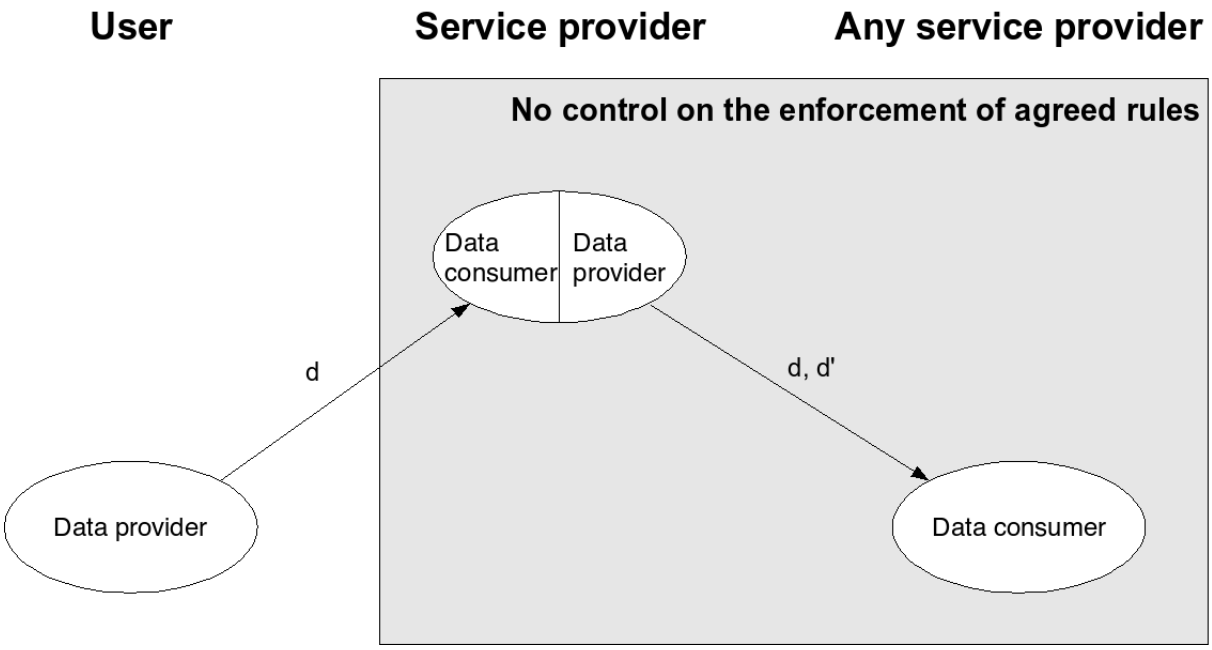

Figure 1 Current, one-sided trust model according to the model of (Pretschner, Hilty and Basin, 2006).

2.2 Extending the one-sided Trust Model

The objective of this deliverable is to extend the one-sided trust model so that users need not to trust service providers regarding the enforcement of the agreed rules. Users should be able to verify whether these rules have been enforced (cf. Figure 2). Since obligations cannot be enforced by the access control mechanism deployed at the user (Pretscher, Hilty and Basin, 2006), e.g. identity management systems of type 3 (Bauer, Meints and Hansen, 2005), but observed, this study focuses on monitoring the usage of personal data according the agreed obligations. It has to be assured that such a monitor is deployed at the information system of the corresponding service provider. A widespread approach of the industry for giving such an attestation is Trusted Computing, e.g. as it is defined by the specification of the Trusted Computing Group (TCG) (Trusted Computing Group, 2003b).

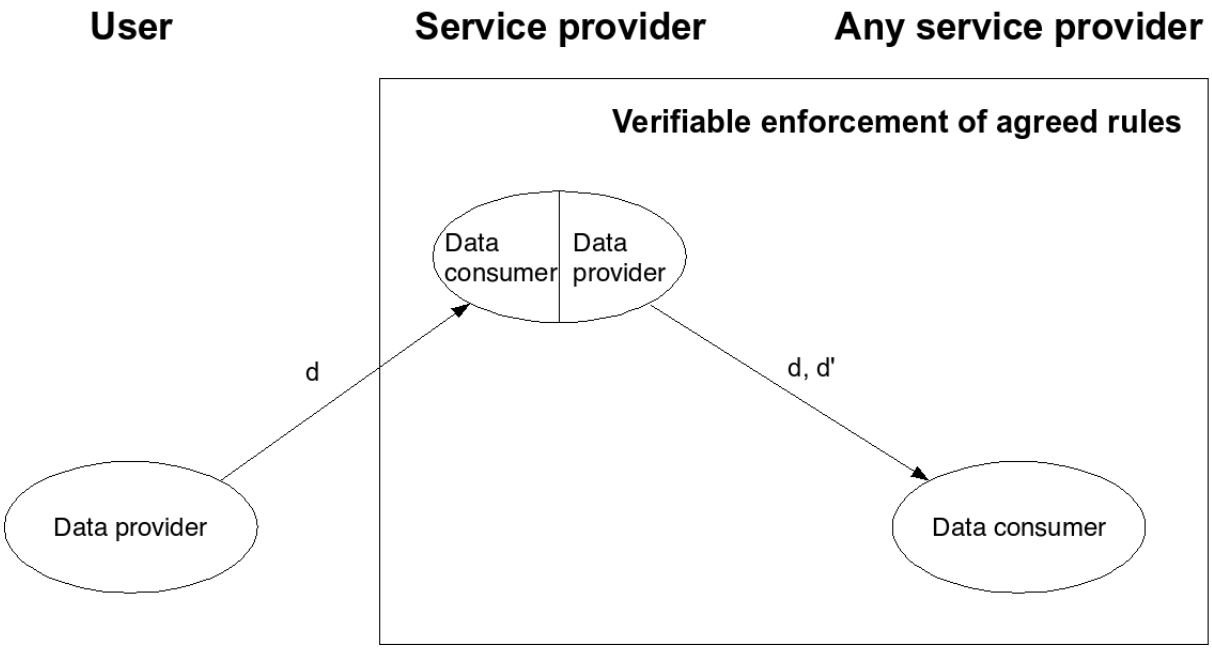

Figure 2 Service providers show their trustworthiness.

2.3 How to read this Deliverable

This deliverable distinguishes between certified and not-certified service applications. "Certified" means that a service application is only able to use personal data in compliance with the agreed rules (Zugenmaier and Hohl, 2004). The challenge in this case is to give users the possibility to prove whether personal data is sent solely to a certified application. By the authentication of a service application instead of a service provider, the service provider is able to guarantee that personal data will be sent to this specific application. Therefore, this deliverable presents a technical approach in order to disclose personal data to a specific application.

By using uncertified service applications, users have no knowledge about the behaviour of this application and whether it processes personal data in compliance with the agreed rules. The deployment of uncertified applications makes sense if they are modified frequently and the certification take too much effort. In this case, the challenge is to observe and to log the use of personal data and to make the resulting transcript available to the corresponding user. With knowledge about the behaviour of the specific software and hardware components by investigating on logged data according to the access decisions of service provider's information system on queries for personal data, a user can decide if they are in compliance with a given privacy policy and thereby trustworthy. This deliverable investigates on the realisation of such an observation.

Chapter 3 introduces the requirements for the controllable processing of personal data. It starts with the derivation of the current, one-sides trust model by the use cases "Information filtering" and "Personalised Services". In order to improve the trust relationships and so to realise the trust model where service providers verify their enforcement of the agreed rules, the authors describe the requirements of a monitor for the observation of processing of

personal data. Trusted Computing seems to be a suitable approach for deploying a trustworthy monitor.

Chapter 4 introduces the concept of Trusted Computing by the TCG. Readers who are familiar with the specification of the TCG may skip this chapter.

Chapter 5 investigates on the employment of Trusted Computing in order to support the enforcement of privacy policies. This chapter shows shortcomings of today's Trusted Computing concerning this kind of employment, e.g. the time problem. This problem considers the change of an information system between its attestation and the collection of personal data. An attacker may be able to exchange the attestated service application by his own application and thereby to get personal data. A user cannot recognize this modification by using Trusted Computing base on the specification of the TCG.

Chapter 6 presents approaches for solving the time problem and using the TCG specification in order to verify the behaviour of an information system by a monitor. Such a monitor encapsulates the service application of a service provider in order to observe the usage of personal data according to the agreed obligations. As one premise for this use of Trusted Computing, this chapters presents a solution for solving the time problem which is introduced by chapter 5. Afterwards, two approaches for monitoring privacy rules by using Trusted Computing are presented. The first approach considers an encapsulation of a service application by a monitor in order to detect an information flow of personal data which violates the privacy rules. The second approach presents the protection of private data in th use case “Information Filtering” by using Trusted Computing.

Chapter 7 summarizes this investigation on the use of Trusted Computing for enforcing privacy policies by its results.

3 Business Processes and a verifiable Processing of Personal Data

This chapter investigates on the scenario and its flow of personal data in business processes for personalised services. Two use cases illustrate this model and show the demands on a verifiable processing of personal data. The trust model focuses on the confidentiality of personal data and on the accountability service provider's activities on personal data. From this trust model, requirements for a verifiable processing of personal data are derived. Users are not able to monitor the usage of their data, since these activities happen in the information system of the service provider. So, the required security mechanism has to be deployed at the information system of the service provider but hast to be controlled by the user.

3.1 Information Flow Model of Business Processes for Personalised Services

Business processes realising personal services are characterized by a collection, usage, storing and delegation of personal data. Storing and using personal data, e.g. for marketing purposes, are thereby implied. The actors are *data consumers* and *data providers*. A data provider stores personal data in a trustworthy place and decides on access on this data, whereas a data consumer requests personal data. An actor sometimes changes his role depending on the service from a data consumer to a data provider. Beside the access on personal data, its usage has also to be considered, e.g. its exchange between service providers. If services of a personalised service form a chain of services, users need also to control the delegation of personal data, if informational self-determination should be preserved by technical means. Business processes with such a chain are called multi-stage business processes. Figure 3 shows the information flow model of multi-stage business processes. A data provider delegates personal data *d* to a data consumer, who further delegates this data together with additional data *d'* in his role as a data consumer to another service provider, here as a data consumer (Pretschner, Hilty and Basin, 2006).

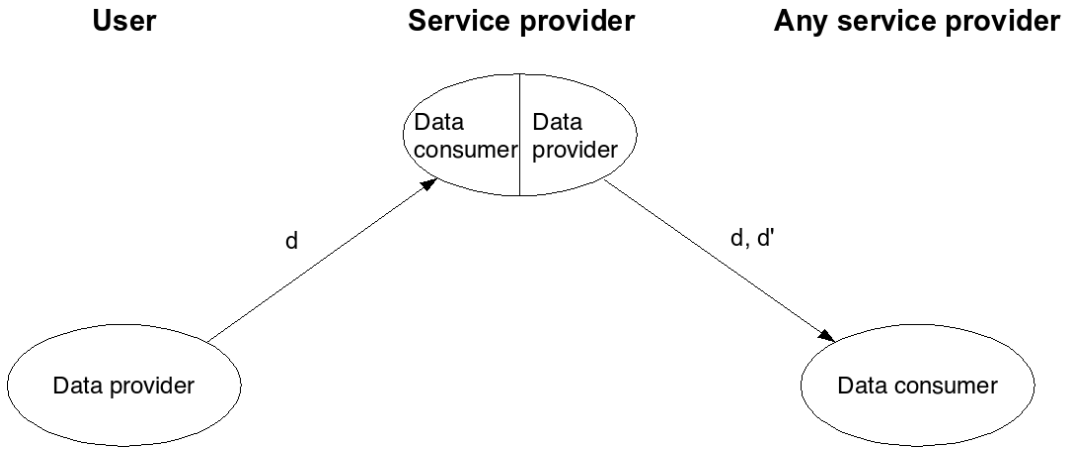

Figure 3 Information flow model of multi-stage business processes (Pretschner, Hilty and Basin, 2006).

This information flow model and the dynamic roll change are explained by the use case "Loyalty Programme" as described in (Müller and Wohlgemuth, 2007). Concerning a collection of personal data, a service provider acts as a data consumer towards his users and as a data provider towards the loyalty programme provider. In this case, the loyalty programme provider takes the role of a data consumer. In case of delegation of personal data, the loyalty provider takes the roles of a data provider and the corresponding service provider takes the role of a data consumer (cf. Figure 4).

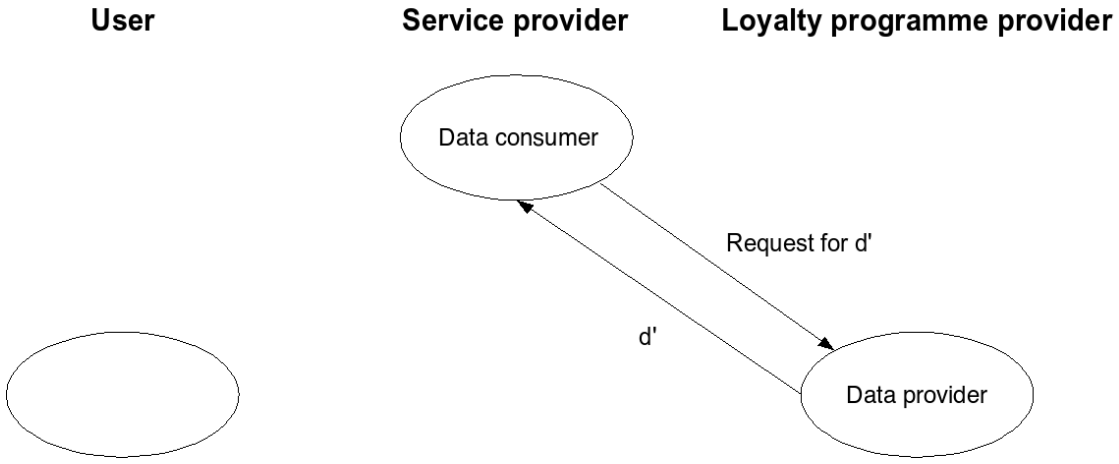

Figure 4 Actors and their roles in a loyalty programme, if personal data is being delegated (Müller and Wohlgemuth, 2007).

3.2 Use Case: Information Filtering

Information filtering systems aim at countering information overload by extracting information that is relevant for a given user out of a large amount of information available via an information provider, based on user profiles containing, for each given user, personal data such as preferences, and/ or rated items. Information filtering systems may be applied in order to generate recommendations of items that are probably relevant for a given user, in order to predict the relevance of specific items for a given user, or in order to determine users with similar interests. Typical examples are the recommender system used by amazon.com which suggests potentially relevant books or other multimedia, based on the respective user's preferences and shopping history. Another example is the recommender system used by last.fm which provides personalised song play lists. While in these well known cases the information provider and the provider of the recommender system functionality are realized by a single entity, information filtering systems in most cases constitute a multi-stage business process in which personal data has to be delegated from the information provider to the recommender system provider (cf. Figure 5). The information *d* in this case is the result data, e.g. a set of recommendations.

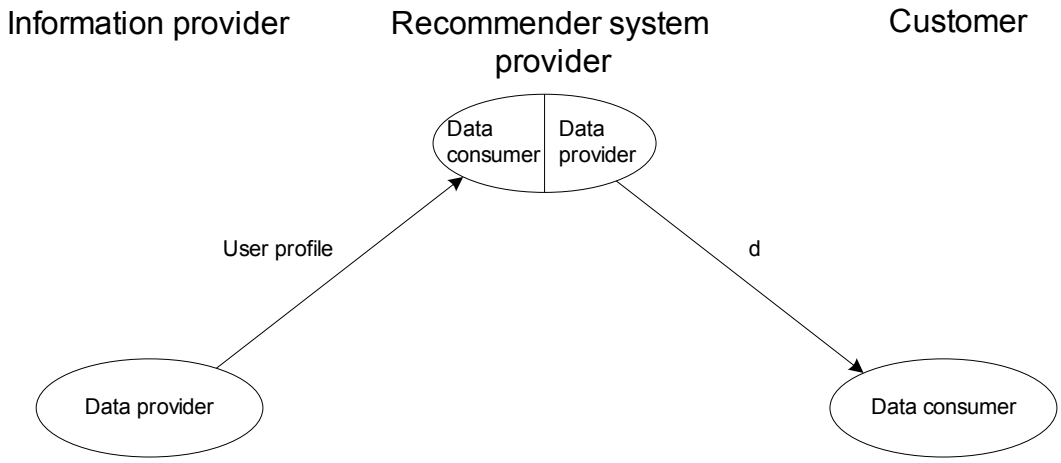

Figure 5 Actors and their roles in an information filtering system, if a user profile is going to be delegated in order to offer personalised information, namely recommendations.

In information filtering, the problem introduced above arise because the information provider as well as the recommender system provider may use or delegate the user profile without the consent of the user and thus violate the agreed privacy policy. Users are therefore often reluctant to use recommender systems especially in the context of sensitive information domains, where the user profile contains e.g. financial or health-related personal data.

3.3 Use Case: Personalised Services

In personalised services, the service provided to the customer is based on pre-acquired personal information around the customer himself. In many personalised services the service provider obtains customer information from other data providers. Therefore, customer information following this information flow model is subject to disclosure at several points.

The problem originating from disclosure of private customer information could be considered a main drawback in personalised services. The personal data provided by the customer can be intercepted or misused in several ways. In fact, the confidentiality of the personal data can be breached not only through man-in-the-middle attacks, but also through leakage of information from within the service provider himself. Accountability of the service provider is in question here, in addition to its technical and managerial ability to abide by the privacy policy. In the case where a customer profile is delegated to the service provider, the privacy policy might be agreed upon only with the first data consumer to which the personal information has been provided. For these reasons, we consider in the following a use case of personalised services to shed the light on the flow of personal data in corresponding business processes.

We consider the case of a company “A” providing to its client bank “B” fully integrated back office processing and personalised services for B’s customers. A is able to track customer information across multiple channels in a way to enhance B’s customer care, acquisition and support. The personalised services provided by A on behalf of B include customized internet-banking applications, online account management applications, telephone banking services, etc.

The collection of the customer's personal information is done by B as part of the customer registration process. Encryption schemes and access control systems could be used on B's information systems to ensure the confidentiality of the customer private information. Moreover, a privacy policy is usually agreed upon between B and the customer regarding the use of private data.

However, the customer's profile information has to be delegated to A in order for the latter to provide the personalised services. At this stage, the specific customer information passed to A can include account information, transactions history, withdrawal limits, credit card information, telephone numbers, etc. The profile information delegated to A should be restricted to the necessary information fields needed to provide the corresponding personalised service. Otherwise, the purpose-binding of the profile would be lost. This resulting information can be further abused by A, e.g. for advertisement purposes in the favour of A, and that have no relation to the banking service delegated by B. For example, A could use the telephone number to perform some survey on the corresponding customer which is necessary for other services that A can provide. But A would be more inclined to avoid such privacy breach in order to protect its and B's reputation.

On the other hand, leakage of the customer information from A's information system should also be avoided, since A would become accountable for any customer private information disclosure. For example, a leaked credit card billing address can be used for advertisement purposes which address directly the corresponding customer. Encryption systems can also be used by A to control the access to private customer information.

Although the privacy policy agreed on between the customer and B at the time when the private information was provided should protect the privacy rights of the customer, the enforcement of this policy is not trivial in the case of a multi-stage business process, especially when personalised services are offered by a partner enterprise. Using access control mechanisms to prevent a breach of the privacy policy is not enough, since the partner enterprise personnel will definitely have access to the customer private information in order to provide the banking service. On the other hand, using logging and auditing of security critical actions in order to validate the enforcement of the privacy policy after the privacy breach took place would not thwart the consequences of a credit card number theft, or a telephone banking impersonation, although the partner enterprise would be held accountable.

A controllable enforcement of privacy policies by a monitor is a preventive approach. This requires a continuous run-time monitoring of the customer private information usage in a way to prevent misuse of information according to the privacy policy as opposed to controlling the access to the information. Enterprise Rights Management (ERM) technique could be a good solution to enforce such a privacy policy. With ERM (Enterprise Strategy Group, 2006), different roles within the partner enterprise have different fine-grained usage rights over the information. In addition to an access control model, an ERM solution can help controlling the customer information usage by making some information fields (credit card numbers, online-banking passwords) accessible to particular persons or roles within the partner enterprise providing the banking service only.

3.4 Trust Model concerning Processing of Personal Data

In the deliverable D14.2 "Study on Privacy in Business Processes by Identity management" of FIDIS Work package 14 (Müller and Wohlgemuth, 2007), the trust model focuses on service providers and on the privacy threat of an undesired profiling while personal data is collected and delegated. The trust model of this deliverable focuses on service providers and their activities relating to the processing of personal data. A user cannot be sure whether service providers follow the agreed privacy policy.

Privacy threats arising by the attacks consider an undesired collection, storage and delegation as well as a misuse of personal data. If personal data has been collected by others than the addressed service provider, e.g. by impersonating him by means of a man-in-the-middle-attack, the confidentiality of personal data is not given anymore. If personal data has been stored, delegated or used contrary to the consent of the user, the confidentiality of his profile compared to the agreed privacy policy has been violated.

The protection goals of multilateral security (Rannenberg, Pfitzmann and Müller, 1999) are the initial point of the trust model. The focus of the attacker model is on the protection goals of "confidentiality of personal data" and "accountability of service providers' activities" relating to the use of personal data. The security criteria "anonymity" and "unobservability" of multilateral security are not considered, since users disclose explicitly personal data to service providers.

Confidentiality of communication with respect to personal data means according to (Rannenberg, Pfitzmann and Müller, 1999):

- The transmitted personal data is only known to its recipient
- The communication partners may be unknown.
- The location of communication partners may be secret.

The first criterion is of relevance for personalised services, but not the second and third criteria, since users disclose explicitly personal data to service providers for the purpose of a personalised service.

Furthermore, a *confidential data processing* means for this study that disclosed personal data is only used for a specific purpose and with consent of the corresponding user. The scope of this usage is pre-defined by a policy agreed by the user and service provider.

Not every service application is able to process personal data confidential and to ensure this quality to its users in advance. In this case, users should be able to verify whether their data has been used according to the agreed privacy policy. It follows that the activities of a data usage must be controllable and accountable. Multilateral security defines the protection goal **accountability** as follows:

- Service providers should prove to a third party that entity X has sent or used the message or service Y.
- A user should be able to prove that he has sent or used a message or a service and in case of messages that he has received a given one.

- Nobody is able to deny fees for supplied services; at least service providers get a proof for supplied services.

In the context of privacy, the first aspect is interpreted as a user is able to verify whether a service provider has delegated, stored or used personal data. The other criteria are of lower relevance as they follow a reactive approach. A verifiable data processing does not enforce user's privacy interests but enables him to initiate sanctions in case of a privacy violation.

3.5 Requirements for a verifiable Processing of Personal Data

A security mechanism to support a verifiable data processing with respect to the protection goals confidentiality and accountability must have the following properties:

- Identification of a collection of personal data contrary to the agreed privacy policy.
- Preventing storage and delegation of personal data which is in conflict to the agreed privacy policy or at least generating an unmodifiable transcript of these activities.
- Preventing an undesired usage of personal data, that means user has not given his consent to this usage, or at least generating an unmodifiable transcript of such activities.

Proving the authenticity of service providers is a countermeasure against man-in-the-middle attacks, but it is not sufficient for protecting against a misuse of user's profiles. The behaviour of the information system of service providers remains unknown to the user. Therefore, an authentication addresses not a service provider but his information system. In case of a certified information system, a user gets an evidence for its possible behaviour. In case of an uncertified information system, the security mechanism must create evidences for the behaviour of the information system concerning the collection, use, storage and delegation of personal data. Creating an unmodifiable transcript is one step towards such evidence. Since those activities require access to personal data, the monitor of the access control system of service providers has to be modified in order to log these activities. **A user must be able to verify whether this monitor is part of the information system and whether applications concerning the processing of personal data are controlled by this monitor.**

Trusted Computing is a technique for ensuring the authenticity of software, e.g. such a monitor. In the following, this technique is introduced and evaluated for this purpose and a monitor is presented in order to support the enforcement of privacy policies.

4 An Introduction to Trusted Computing

In this chapter, the basics of trusted computer platforms (TCP) and the specification of the Trusted Computing Group (TCG) are presented. It is in relation with FIDIS deliverable D3.9 on the impact of Trusted Computing on identity and identity management (Alkassar and Husseiki, 2007). If the reader is already familiar with the basics, this chapter can be skipped and reading continued on the application of TCP.

It is anticipated that the functionality of trusted computer platforms will form part of the basic configuration of the most varied kind of computers within a few years. Numerous enterprises, such as the processor manufacturers ARM (ARM, 2004) and Transmeta (Transmeta Corporation, 2003) and consortia like the Trusted Computing Group (Trusted Computing Group, 2003b) are developing specifications and architectures for personal computers, PDAs, mobile phones and embedded systems.

Trusted Computing technology is a relatively new technical approach aimed at providing evidence about the integrity of a platform to both, the platform's owner and to arbitrary third parties. The degree of confidence in software-only security solutions depends on their correct installation and execution, which can be affected by other software that has been executed on the same platform. Therefore a trusted platform is a conventional platform containing a hardware-based subsystem devoted to maintain trust and security between machines. It contains a trusted component, probably in the form of a built-in cost effective security hardware that is used to create a foundation of trust for software processes. This extra hardware is roughly equivalent to that of a smart card (with some enhancements) and contains a variety of functions that must be trusted. Although the first publications around Trusted Computing date back to the 1970s, the Trusted Computing Group has boosted the research and development efforts in this field.

The TCG evolved from the Trusted Computing Platform Alliance (TCPA) which was an industry working group focused on the development of trust and security mechanisms in computer platforms. It was formed by Compaq (today part of Hewlett-Packard), Hewlett-Packard, IBM, Intel and Microsoft in January 1999. In August 2000 the first public version of the TCPA Specification was released for comments and has been published as TCPA Specification 1.0 in February 2001. This specification was platform independent and basically defined functions that must be provided by a Trusted Platform Module (TPM) from the viewpoint of a hardware manufacturer. In April 2003, the TCPA was transformed into a non-profit organization called Trusted Computing Group (TCG) (Trusted Computing Group, 2003a). The TCG adopted all TCPA Specifications and continued their development. In November 2003 the last major change to the TCG Specification has been published as TPM Main Specification 1.2 (Trusted Computing Group, 2003b). It essentially describes the platform independent functionality that must be provided by a TPM. Today the TCG has more than 120 members, including component and system vendors, software developers and network and infrastructure companies. The following introduction to Trusted Computing respectively TPM is based on the specification (Trusted Computing Group, 2003b).

4.1 Architecture of Trusted Computing by the TCG

The main components of the TCG proposal are the hardware component Trusted Platform Module (TPM), a kind of (protected) pre-BIOS called the Core Root of Trust for Measurement (CRTM), and a supporting software called TCG Software Stack (TSS) which performs various functions like communicating with the rest of the platform or with other platforms.

The TPM Specification is the main part of the TCG Specifications. It defines all platform independent aspects and functions that must be provided by a trusted platform. All system specific aspects have been sourced out to system specific documents like the PC Specific Specification.

The TPM provides an RSA key generation algorithm, cryptographic functions like RSA encryption and decryption, a secure random number generator (RNG), non-volatile tamper-resistant storage, and the hash function SHA-13.

The TCG Specification does not prescribe that TPM devices have to be implemented in hardware but to provide the degree of security arrogated by the TCG Specifications with a purely software implementation may be an infeasible task. Thus most TPM implementations are in hardware.

Hardware TPM devices can be compared to integrated smartcards containing a CPU, some memory, and special applications. The assumption is that the chip is tamper-evident and mounted on (or integrated in) the motherboard such that removal is evident to visual inspection. The main chip contains a special security controller with some internal, non-volatile ROM for the firmware, non-volatile EEPROM for the data and RAM. Furthermore, it contains a cryptographic engine for accelerating RSA encryption and decryption processes, a hash accelerator and a random number generator that is needed to generate secure cryptographic keys. The Figure 6 shows the main components of the chip.

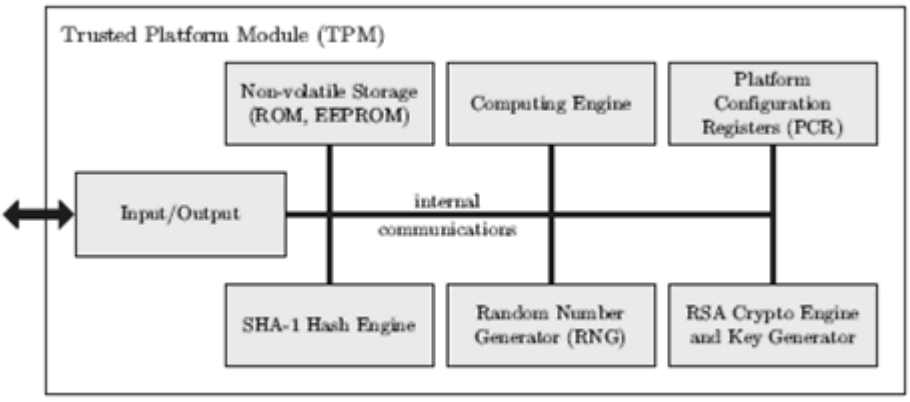

Figure 6 Architecture of the TPM.

A TPM contains a Root of Trust of Storage (RTS) which protects data and keys entrusted to the TPM. The RTS manages a small amount of volatile storage inside the TPM device that is used to hold currently used keys (key slots). Unused keys may be encrypted with a storage

key and moved off the TPM chip, e.g., to a hard disk drive. The storage key might be encrypted with another storage key which leads to a key hierarchy with the Storage Root Key (SRK) being the root. The key slots of the TPM are managed by a trusted service outside the TPM which is called Key Cache Manager (KCM).

4.2 Key Types

Each TPM protected key is stored with several attributes that identify the type of the key and what it is intended to be used for. These attributes are set during the generation of the particular key and cannot be altered later.

Storage Root Key The Storage Root Key (SRK) is used to wrap TPM protected keys which can be stored outside the TPM. This builds a hierarchy of keys on an external storage device like a hard disk drive. The SRK is embedded into the TPM and is generated during the process of creating a platform owner. It can be re-generated by creating a new platform owner which destroys the previous key hierarchy and all the keys it contains.

Endorsement Key Each TPM device is shipped with an embedded non-migratable Endorsement Key (EK) that has been generated as a part of the manufacturing process in or outside the TPM. Embedded means that the key cannot be removed from the TPM and thus uniquely identifies it and the surrounding platform. The entity that generates the EK issues an Endorsement Credential which should provide evidence that the EK has been properly created and embedded into a valid TPM. Besides the two special keys described above, a TPM can create four different types of asymmetric keys:

- Migratable keys (MK): Migratable keys are cryptographic encryption keys that are only trusted by the party who generated them (e.g., the user of the platform). A third party has no guarantee that such a key has indeed been generated on a TPM.
- Non-migratable keys (NMK): Contrary to a migratable key, a non-migratable encryption key is guaranteed to reside in a TPM-shielded location. A TPM can create a certificate stating that a key is an NMK.
- Certified-migratable keys (CMK): This type of encryption key, introduced in version 1.2 of the TCG specification, allows a more flexible key handling. Decisions to migrate and the migration itself are delegated to two trusted entities, chosen by the owner of the TPM upon creation of the CMK: The Migration-Selection Authority (MSA) controls the migration of the key, but does not handle the migrated key itself. In contrast, the Migration Authority (MA) handles the migration of the key:

To migrate a CMK to another platform, the TPM expects a certificate of an MA stating that the key to be migrated can be transferred to another destination. Furthermore, the certificate of the CMK that the owner/user uses to prove that it was really created by a TPM contains information about the identity of the MA resp. MSA.

***Attestation identity keys (AIK)*:** These non-migratable signature keys provide pseudonymity resp. anonymity of platforms including a TPM. AIKs are locally created by the TPM. The public part is certified by a Privacy Certification Authority (Privacy CA) stating that this signature key is really under control of a secure TPM. In order to overcome the problem that this

party can link transactions to a certain platform, version 1.2 of the TCG specification defines a cryptographic protocol called Direct Anonymous Attestation (DAA) (Brickell, Camenisch and Chen, 2004), eliminating the Privacy CA. AIKs can be used to attest to specific platform configuration states. A platform can have multiple AIKs to avoid correlation of platform identities. In order to generate AIKs, the Endorsement, Conformance and Platform Credentials (which are delivered with the platform), the EK and the authorization by the platform owner to use them is required.

4.3 TPM Credentials

A trusted platform is delivered with several credentials (digital certificates) that should provide assurance that its components have been constructed to meet the requirements of the TCG Specifications.

Endorsement Credential As already mentioned, the Endorsement Credential should provide evidence that the EK has been properly created and embedded into a valid TPM. It is issued by the entity that generated the EK. This credential contains the name of the TPM manufacturer, the TPM part model number, its version and stepping and the public part of the EK. The EK is a RSA encryption/decryption key pair which is used along with the Endorsement, Platform and Conformance credentials to provide evidence of the platform's identity in a protocol to establish AIKs.

Conformance Credentials The Conformance Credential is issued by an evaluation service (e.g. the platform manufacturer, vendor or an independent lab) with sufficient credibility to properly evaluate TPMs or platforms containing a TPM. It should indicate that the Trusted Building Block (TBB) design and implementation has been accepted according to the evaluation guidelines. A single platform may have multiple Conformance Credentials for multiple TBBs. They typically contain the name of the evaluator, platform manufacturer, the model number and version of the platform, the TPM manufacturer name, TPM model number and version or stepping and a pointer to the location of the TPM and platform conformance documentation. The Conformance Credential does not contain privacy sensitive information or information that can be used to uniquely identify a specific platform.

Platform Credential The Platform Credential is issued by the platform manufacturer, vendor or an independent entity. It should provide evidence, that the platform contains a TPM as described by the Endorsement Credential. The Platform Credential contains the name of the platform manufacturer, the platform model number and version and references to the Endorsement Credential and the Conformance Credentials. The Platform Credential is privacy sensitive since it contains information that can be used to uniquely identify a specific platform.

4.4 Integrity Measurement and Reporting

A trusted platform subsequently collects information about its current configuration and stores it in a log outside the TPM, called Stored Measurement Log (SML). This enables the detection of modified code and malicious or unwanted software which might compromise the platform's security and thus its level of trust. The information stored in the SML cannot be

stored inside the TPM device since it may become very large. Manipulations of the SML will be detected because the digest of the original sequence is securely stored inside the TPM. For this purpose the TPM provides a set of registers called Platform Configuration Registers (PCR) that can be used to store hash values. The TPM hardware ensures that the registers can only be modified as follows: $Ri+1 := SHA1(RijI)$, with the old register value *Ri*, the new register value *Ri*+1, and the input *I*. The process of modifying a PCR value is called extending a PCR and ensures that related values will not be ignored and the order of operations is preserved.

The content of the PCRs can be used for verifiable attestation of the platform's configuration based on Validation Credentials and the chain of trust. Validation Credentials are digital certificates issued by hard- or software manufacturers that provide measurable components (like video and disk storage adapters, memory controllers, processors or software) or other qualified validation entities. They contain the validation entity name, component manufacturer name, component model number, version or stepping and digitally signed reference measurement values taken in a clean-room environment when the component is believed to work properly. The verification of a platform configuration state requires the re-computation of the measurement digest using the reference measurements from the Validation Credentials and a simple comparison of the resulting digest value with the actual content of the PCR. A TPM can attest to a PCR value by digitally signing it with an AIK.

4.5 Chain of Trust

As already mentioned, a trusted platform subsequently reports integrity measurement information to the TPM. The idea is that each firm- or software component that is to be loaded or executed is checked before it is started. The result of this check, a message digest, is reported to the TPM in a cryptographically secure manner. Once the value has been submitted to the TPM, it cannot be changed. This means, that any change or manipulation of the software state can be recognized since malicious software cannot hide itself by manipulating PCR values or the SML. This implies that the instructions that start the chain of measurements must be trusted which means that they have to function as expected. These instructions are called Core Root of Trust for Measurement (CRTM). Ideally the CRTM would reside in the TPM to profit from its tamper-resistance but due to architectural requirements of the specific platform it might also be located in another device (like the BIOS of the PC platform) which can hardly be manipulated from a remote adversary and should be trusted. After the CRTM measured the system environment consisting of firmware and other components required to give control to the platform's computing engine, which typically consists of the system's CPU, memory and chipset, the CRTM passes control to the Root of Trust for Measurement (RTM). Typically the RTM actually is the platform's normal computing engine which has been previously checked by the CRTM. The RTM inherently generates reliable integrity measurements and reports them to the TPM device building a "chain of trust" as presented in the figure below. The term trust means that the software respectively hardware module is correct and thereby behaves as expected.

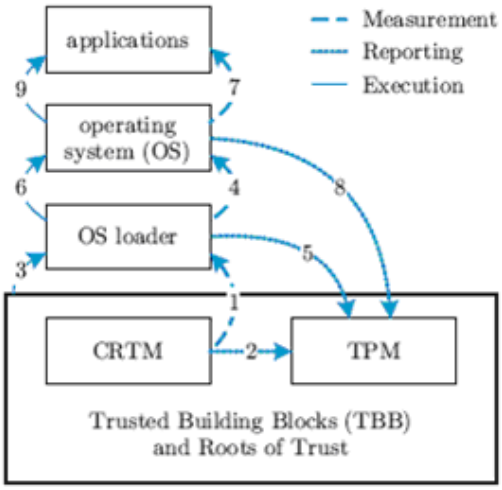

Figure 7 Chain of trust.

Binding
Binding means that a message can be bound to a certain TPM (and platform) using encryption. When encrypting a message with an asymmetric encryption scheme, the sender uses the public key of the recipient to encrypt a message. The recipient is then able to decrypt the cipher text with his corresponding private key which can be managed by a TPM. If this private key is a non-migratable key then only the TPM that generated it is able to use the key and thus decrypt the message. Therefore the message is bound to the TPM that protects the corresponding private key.

Signing
By signing a message, the integrity of this message is associated with the key used to generate the signature. This means that a verifier can detect manipulations of a (signed) message and is able to identify its origin by the verification key which might be bound to an identity using a digital certificate. The TPM tags some managed keys as signing only keys. Those keys are only allowed to be used for signature generation. This should prevent them from being used as encryption keys which might comprise security.

Sealing
Sealing is an extension of binding since sealed messages are additionally bound to a set of platform metrics specified by the sender of the encrypted message. These metrics describe a specific platform configuration state that must exist before the decryption key is allowed to be used and thus the encryption of the message is possible. Therefore Sealing binds a message to a set of PCR values and a non-migratable key protected by a TPM. This provides assurance that protected messages are only recoverable when the platform is in a specific known configuration which is considered to be trusted by the sender of an encrypted message.

Sealed Signing
Signing operations can be linked to specific PCR values and thus a specific platform configuration state. For this reason PCR values are included into the signature. This enables a

verifier to inspect a platform's configuration at the time when the signature has been generated. The verifier is then able to decide whether to trust the given platform configuration state and accept the signature or not.

Attestation
Attestation is the process of vouching for the accuracy of information. A TPM can attest to information by digitally signing internal TPM data like PCR values using an AIK. The correctness of this information then can be verified by a third party that checks the integrity measurements and the AIK itself. The AIK can be obtained and verified by using a Privacy CA or a trusted attestation protocol like DAA.

5 Employment of the Trusted Computing Platform

The functionality provided by Trusted Computing platforms can be beneficially used by numerous applications. As Trusted Computing technologies are controversially discussed from the point of view of privacy and adverse effects for users (Anderson, 2003a; Anderson, 2003b) on account of their technical implementation, two cases of application are to be differentiated. In the first case, Trusted Computing is employed on the side of the user and in the second case the employment of Trusted Computing technology is limited to the employment on the server side by service providers[1].

In the following, employment on the part of the user is omitted. A series of additional problems from the point of view of the user are to be solved in this application case, but that no immediate advantages for the confidential processing of issued personal data emerge. The employment of Trusted Computing technology on the side of the server is considered in detail. This technology can be used here for the authentication of applications and services to convey valuable information to the user about the service provider and the behaviour of its applications concerning processing of personal data.

5.1 Application Areas of Trusted Computing on the Server Side

The first seminal publications in the field of Trusted Computing (TC) can be dated back to the early 1970s. It became an "emerging"' technology in the past few years due to the fact that an industry consortium — the Trusted Computing Group (TCG) — started to implement the necessary hard- and software parts in order to build a sound and secure Trusted Computing platform. Nevertheless it is still (emotional) discussed what exactly Trusted Computing is[2] and whether it has more benefits for users or for commercial organisations e.g. in scenarios like Digital Rights Management (DRM).

For now we assume that Trusted Computing comprises at least the following technologies and mechanisms:

- **Trusted computing base** which is the minimal set of hardware (e.g. the TPM-chip specified by the TCG), firmware and software (e.g. a secure operating system) which is necessary for enforcing a security policy.
- **Secure I/O** which offers protected paths for all data from the input over the processing until the output. That means for instance that the user can be sure that the inputs he made can not be intercepted by malicious software like keyboard loggers.
- **Sealed memory** which refers to a protected memory which prevents other processes (and even unprivileged parts of the operating system) to read/write to it.
- **Sealed storage** a technology which ensures that persistent data can only be read and modified by exactly the same combination of hard-/software which has written the data.

[1] This does not signify that no Trusted Computing functionality may be present on the side of the user, but that these are merely irrelevant in this context.

[2] This is not surprisingly as the term "trust" itself is heavily discussed within different communities.

- **Authentic booting and (remote) attestation** which allows a user to be sure with which well defined hard-/software he interacts and to prove this even to third parties.
- **Unique digital identities for computers** which means that each Trusted Computing base has a unique digital identity enabling the owner of a computer to prove that a certain message originated from a computer he owns or that two messages come from the same computer; that two messages do not come from the same computer.

An important fact and fundamental principle about Trusted Computing is, that Trusted Computing does not mean, that the computing environment (hard- and software) can be trusted—but instead one *has to* trust it. According to Ross Anderson, "In the US Department of Defense, a 'trusted system or component' is defined as 'one which can break the security policy'."[3] This simply means, if the trustworthiness assumptions one has about a certain Trusted Computing based ICT system are wrong, then the whole protection offered by this system (in terms of security and privacy) can be broken.

Immediately the question arises to what extend one should trust the Trusted Computing. If one is willing to absolutely trust the Trusted Computing many (if not all) security and privacy related problems can be solved very easily. The reason is that most of the complex and complicated cryptographic mechanisms and protocols just exist or were designed with the goal to circumvent the untrustworthiness of the computing environment (soft- and hardware) used by the communication partners and third parties. As example FIDIS deliverable D14.2 identified the delegation of rights and secrets to proxies which act on behalf of the customer as one of the fundamental problems (with respect to security and privacy) in multi-stage businesses processes. Clearly if these proxies are not trustworthy, then they can use the data provided by the customer to contravene the interests of the customer and violate his privacy. Using Trusted Computing on the proxy side could easily solve this problem (under the assumption that one is willing to absolutely trust the Trusted Computing as mentioned above). In this case the proxy would be trustworthy (and can be trusted) "by definition".

On the other hand the history of security and privacy technologies as well as ICT in general has shown, that such absolute error-less and correct designed and working systems does not exist and will (with high probability) never exist. Therefore Trusted Computing should only be seen as a "helping tool" which could be used to enhance the overall security a system provides.

[3] Ross Anderson: 'Trusted Computing' Frequently Asked Question. Version 1.1 (August 2003), http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html

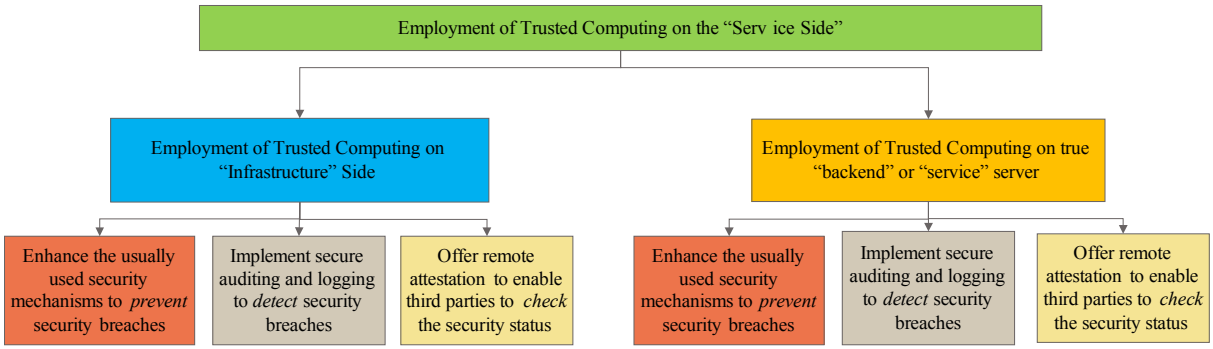

Figure 8 Overview of approach to "employment of Trusted Computing on the service side".

In (Iliev and Smith, 2005) the fundamental property of Trusted Computing is described as follows: "We call the physically protected and trusted components of a server K, ... In any given client-server application, we can view K as an extension of the client: from a trust perspective, K acts on the client's behalf, but physically, K is co-located with the server."

Derived from this fundamental property, using Trusted Computing on the service side comprises at least the following three overall goals /approaches:

- To *prevent* security threats by implementing (traditional) security mechanisms in a more trustworthy way or (more general) use Trusted Computing to secure the basic operations of the servers used on the service side. This comprises all the well known technologies offered by Trusted Computing.

  As mentioned above the communication between the involved parties (users and services) has to be confidential and integer. The (cryptographic) protocols and measures used can benefit from TC and the TPM used on the service side, e.g. the cryptographic keys could be stored under the control of the TPM (using the Sealed Memory and Sealed Storage functionality) making attacks on the communication confidentiality much harder.

  In general it seems that this "classical" approach for enhancing the security on the service side is the one which is in the focus of the industry and corresponding business consultancies[4].

- Enable third parties to monitor and audit the activities of the service's servers and infrastructure in a trustworthy way to *detect* security or privacy breaches. As the necessary log entries might contain sensitive data (e.g. personal data) as well it needs to be specified who has access to which log entries under which circumstances. Moreover the log entries should be anonymised or pseudonymised so that any unnecessary sensitive data is removed. The access rights could be specified in a policy. Trusted Computing could again help to enforce this policy.

  Secure logging is in general a hard task. This is especially true, if the logs should be used for external audits, i.e. if a third party should be convinced and confident that the

[4] See for instance: Jon Oltsik: "Trusted Enterprise Security. How the Trusted Computing Group (TCG) Will Advance Enterprise Security." White Paper, Enterprise Strategy Group, January 2006, https://www.trustedcomputinggroup.org/news/Industry_Data/ESG_White_Paper.pdf

logs were not manipulated. The reason for this is based on the attacker model for secure logging: The one who produces the log is the attacker with respect to integrity and accountability. Basically the logging procedure can be divided into two subtasks: creating the log entries and storing them. In order to enhance the security and trustworthiness of the second subtask this task could be outsourced to a third party which does not collude with the party which generates the log entries (i.e. with the attacker). Sealed storage could be used to bind the log data to a trustworthy state. This would prevent manipulation of the log later on, too.

But securing the first subtask is much harder: if one has full control of the whole system it is impossible to prevent a malicious party from manipulating the generation of log entries (e.g. suppressing or changing them).

Here Trusted Computing can help to solve that problem. Fundamentally it takes away control over some parts of the system from the service provider. One possibility is that the party which is responsible for the audit installs a "trusted log entry generator" (e.g. a log entry generator based on Trusted Computing) within the system of the service provider. If the service provider tries to manipulate this device it would stop logging at all—giving the auditor evidence that some manipulation happened. Moreover the trusted log entry generator could also be verified by third parties by means of remote attestation. This would imply that even the end user (client) could check if the logging done at the service provider is trustworthy from the user's point of view.

Note that it is not sufficient that only the pure log entry generator is under control of Trusted Computing – in addition any part of the system which usually would generate events which trigger the log generator to create a new log entry have to be under the control of Trusted Computing as well (otherwise the attacker could simply suppress the generation of that triggering events). If for instance personal data is stored on a hard disk and access to these personal data should be logged then no ways of circumventing the triggering of log entry generation must exist (i.e. it must be impossible to remove the hard disk from the system and read it using another (non monitored) machine). In case of the hard disk example Sealed Memory in combination with Sealed Storage and Secure I/O can achieve this.

- Enable third parties to *check* the security status of the service infrastructure and servers. Trusted Computing (and especially the remote attestation feature of it) would allow any third party to check whether the systems used by the service provider are in a trustworthy state or not. Here "trustworthy" means that the systems are in a state which enables them to enforce the security or privacy policies promised by the service provider.

  Although it is possible (in theory) that the end user will perform this check, in practice it will require very complex verification procedures (especially in the case of multi-stage business processes). Therefore it seems to be more feasible that a trusted third party will do this (on behalf of the user). Data protection authorities could be such trusted third parties, which could also issue some kind of seals to the service providers. Of course these checks have to be repeated on a regular base. But as they could be done mainly automatically the overall effort would be within reasonable limits.

Note that with respect to the last two methods the service provider can also take over the role of the "client", i.e. he does not only offer that third parties can detect security breaches or check the security status of the systems used—but he also uses these offers from other third parties to which he has delegated parts of the business process to check whether they comply to their security and privacy policies or not.

Besides these three basic principles applying Trusted Computing on the service side can be further divided into:

- applying Trusted Computing on the *infrastructure* side
- applying Trusted Computing on the real back-end or other service related *servers*

Besides the components (servers etc.) which are obviously part of the overall system used by the service provider parts of the system might exist for which it is not so obvious that they need to be included in a security analysis. Below we will give a few examples for this infrastructural side of the system used by the service provider. We show how this infrastructure (which enables or supports the overall business process) can be secured by the usage of Trusted Computing, too.

On the one side, RFID based systems are seen by many analysts as a key enabling technology for optimising the logistics and the supply chain management. On the other side the security weaknesses of today's RFID systems could introduce a security risk (or even a violation of) the security and privacy policies given by the service provider to the user. Imagine for instance the typical scenario where a service provider chooses a third party for package and shipping. If the RFID tags used for these logistics (e.g. RFID based address labels instead of barcodes) are insecure (e.g. could be read by any third party without proper access control) this would violate a privacy policy stated by the service provider, which expresses that no unauthorised third party will ever learn the personal data provided by the customer.

A general overview about security and privacy mechanisms for RFID based systems is given in the FIDIS deliverable D12.3 "A Holistic Privacy Framework for RFID" (Fischer-Hübner and Hedbom, 2007). Therefore we concentrate here on the possible solutions which use Trusted Computing (Molnar, Soppera and Wagner, 2005). The protection goal is the confidentiality of the data transmitted by a RFID tag to the RFID reader, whereas the attacker model assumes that the RFID reader is under control of the attacker. Trusted Computing (especially the remote attestation and sealed storage mechanisms) may be used within the RFID reader to take away the control over some parts of the reader from the attacker. In general the RFID reader is split into three parts: the Reader Core, a Policy Engine and a Consumer Agent. The Core provides the basic functionality of the RFID reader and has to be small enough so that the integrity measures of Trusted Computing are feasible (e.g. secure booting, secure operating system etc.). The Policy Engine is responsible for enforcing the security policy. This security policy is stored within the Policy Engine and determines which tags an RFID reader is permitted to read and which operations are allowed on the read data (e.g. transmission to the back-end system). Moreover the Policy engine controls access to all secret keys need for communicating with RFID tags (e.g. for confidentiality or authentication). The Consumer Agent enables third parties to audit the activities of the RFID reader. It logs all relevant operations of the RFID reader (like reading tags, accessing secret keys, transmitting data etc.) and makes these logs accessible by third parties. These third

parties can use remote attestation to verify that the desired Consumer Agent is running on the RFID reader. Moreover if the Consumer Agent detects that the Reader Core or the Policy Engine were compromised it will stop the operation of the Reader and report this accordingly.

The given RFID example shows once more the application of the three basic principles mentioned above: to *prevent* security threats, to *detect* security breaches and to allow third parties to *check* the security status. Moreover it underlines the need that each security or privacy relevant part needs to be under control of Trusted Computing if the overall system the service provider uses to provide his service should be secure even against the owner (i.e. the service provider).

Another field of application of Trusted Computing on the service side is related to the infrastructure which is used to support or enable the overall business process. Often parts of the infrastructure or of the processing of the data are outsourced to third parties. This could comprise "classical" procedures like renting servers from providers or giving data to a third party which does the processing and returns the desired results. In both cases typically bilateral agreements are concluded which (e.g. in terms of security and data protection) regulate the various duties and responsibilities. In future emerging technologies like Grid-computing (e.g. for "renting" computing power) or P2P based systems might be involved as well. These systems are much more decentralised and it is much harder to give any guarantees regarding security or data protection (Gasson and Warwick, 2007).

Basically one can see this "outsourcing" as some kind of "low level" delegation compared to the more "high level" delegation in case of multi-stage business processes. Therefore the same general mechanisms (as described above) could be applied. The service provider could for instance use Trusted Computing to ensure that the data processing on a rented server is under his exclusive control and that the provider who owns the hardware has no access to the data. He can use secure logging and audit in conjunction with remote attestation to verify that the machines involved in a large Grid will not leak any sensitive data.

5.2 TCP Employment on the Server Side

While TCP employment on the user side mainly entails drawbacks for the user, the employment of TCG platforms on the side of a service provider (see Figure 9) offers several benefits for both sides. With the employment of TCG platforms on the side of the service provider, users can obtain additional information about its state. Since this technology correctly communicates information about a system and the service user is recipient of additional data there are no drawbacks for him as with the employment on the user/client side and the service user is in a position to make an assessment of the system on the basis of this data. The service provider can use TC platforms to increase the trustworthiness and acceptance of his services. Since a provider, in contrast to a user, appears publicly anyway, it can be assumed that he has no interest in protection against an identification of his platform. Due to this line of interest, the problems and concerns about TCG platforms do not pose any more barriers in this connection.

A user of a service operated on a Trusted Computing platform can receive further information via this extension, provided that the operator of the platform permits it. The accountability of the operator can thereby improve from a user's viewpoint.

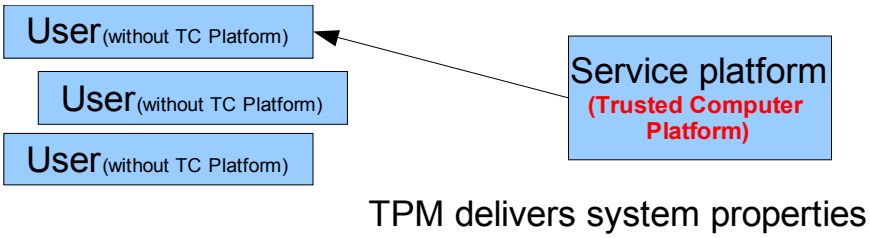

Figure 9 Usage of a TCG platform on the side of the service provider.

The concept of only executing known code or application code of certain manufacturers or certified code of selected third parties has already been proposed by (Rubin and Geer Jr., 1998). However it is only possible to communicate an authentication of a remotely running application to an external enquirer in a trustworthy manner through secure coprocessors or TCP approaches. On the basis of this authentication, an enquirer can then determine whether a specific application can be found on a remotely running platform, with which he wishes to communicate. For platforms complying with the Trusted Computing Group specification, the authentication of applications is carried out by way of remote attestation.

Figure 10 illustrates an attestation of a service platform activated by the user of a service and the subsequent evaluation steps by the user. This attestation is fundamental for the approaches presented in this study, since Trusted Computing will be used for assuring the communication of a user with a monitor for enforcing obligations. Provided that a service provider allows the attestation of his systems, a user has the possibility to authenticate this on the basis of the system state and use this as a further criterion for a utilization of services. The service application can thereby already be known to the service user or it has been certified by a trustworthy third party. This requires that a service provider has to fulfil criteria which have to be laid down, e.g. with regard to the confidential processing of personal-related data in order to be listed with a trustworthy third party. The steps of the attestation are as follows:

1. **Request for an attestation of the platform:** The potential user of a service requests the attestation of the service platform before using the service, i.e. the trustworthy verification of the software components and applications implemented there. The service provider prompts the TP module of the service platform to issue the platform configuration and generate its signature.
2. **Verification of the authenticity of the attestation result:** The user receives the attestation information and examines whether its signature has been generated by a valid TPM. For the signature of the platform configuration, the specification 1.1b provides for a classical signature with the RSA algorithm. The specification 1.2 extends the functionality of the TPM by the group signature procedure *Direct Anonymous Attestation (DAA)* based on zero-knowledge protocols. This type of signature can only be allocated to the group of valid TP modules, but not to an individual module. If the signature is valid, the evaluation of the signed data of the platform configuration can be continued. This step is omitted in the Figure, since no communication between the user and service providers.

In the event that the platform configuration has been signed by an AIK issued by a Privacy CA, this would mean examining whether the certificate pertaining to the signature was issued by a trustworthy entity and whether the certificate still has validity or has been revoked. Conversely, the verification can take place through DAA without contacting a certification authority. The platform configuration is likewise signed by a key pair generated in the TPM. It can be substantiated with the DAA procedure that this key stems from a qualified TP module.

3. **Evaluation of the platform configuration:** In this step, a mapping is to be made between the platform configuration and the proven attributes of the service application. The desired service application is thereby to be identified in the first step via the platform configuration and in the second step; properties relating to confidentiality or confidential processing are to be learned about. This kind of information can already be locally available on account of former interactions or has to be procured via an independent third entity (trustworthy certifier), with which a trust relationship exists.
4. **Service access:** The user starts the service utilization when both the attestation results have a valid signature generated by a TPM and the information itself identifies an application with known properties with regard to privacy.

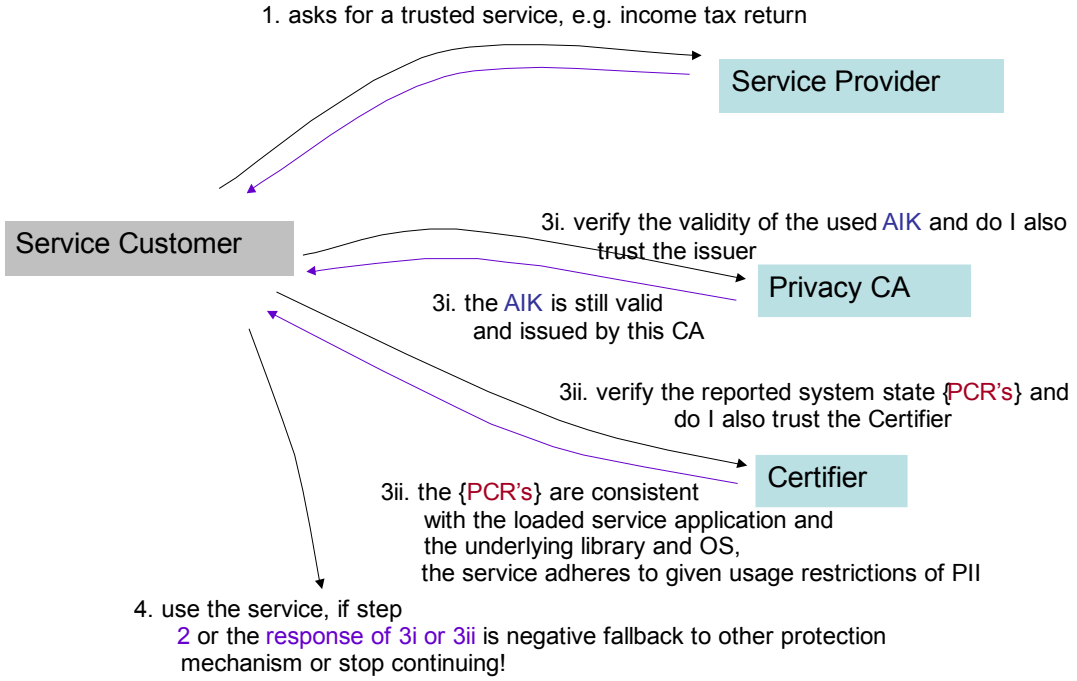

Figure 10 Authentication of the application of the service provider through remote attestation.

The advantage of remote attestation is that the service user can authenticate applications on the service platform. This puts the user in a position to verify the presence of a specific service application with a service platform and to enter into communication with it. This can involve applications which have proven attributes with regard to a confidential processing that are confirmed by a certifier or are transparent to the user himself. Through the employment of

technology, it is proven that a processing of user data concerning verifiable or certified properties takes place. The trust to be placed in the service provider with regard to the processing of user data can be increased by the use of this technology and provide an additional component for the formation of trust.

The transfer from an authentication (of identities) of a service provider to the authentication of his application therefore presents a further decision criterion for the service user when selecting services.

5.3 Shortcomings of TCP for Supporting Privacy

It appears, however, that with the use of TCP and their application on the service provider's side a time-related security vulnerability is present, which can be exploited for a violation of the confidentiality of disclosed data. A substantial part of this chapter is the analysis of this vulnerability and its evasion by means of a technical mechanism.

Expressed in simple terms, trusted computer platforms are approaches with which security attributes of a physical component can be extended to a system and its applications. The aim of TC platforms is to guarantee the integrity of the execution of an application and, if necessary, credibly attest this. To achieve this goal, components (also known as trust anchors) are currently used, which cannot be influenced in their operation by software or physical intervention. Assertions and attestations about a platform are based on these. In addition to the logging of application execution, trusted computer platforms mainly ensure active protection of the applications to be executed by realizing confidential and integer memory areas through the application of cryptography.

TCP applications are thus suitable for giving an outsider credible information about a platform, provided that he is confident that the TCP mechanisms operate as expected. TCP mechanisms can therefore also be regarded as a means for logging and communication and are hence suitable for the transparent execution of remote tasks and responsibilities (see Figure 11).

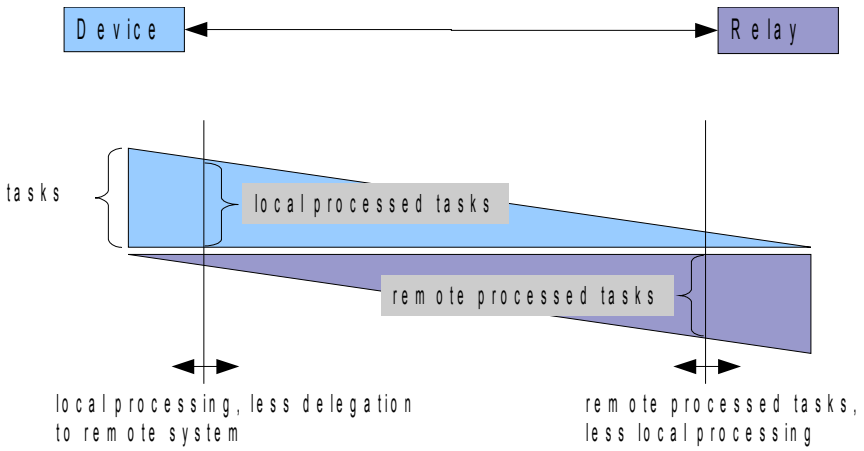

Figure 11 Shifting tasks to remote systems with verification of their orderly execution.

No previously existing trust relationship is thereby required, for example through reputation systems (Cranor and Resnick, 2000; Bohnet and Huck, 2003) between users and providers. This relationship is replaced by the application of additional technology and by trust in the correct mode of operation of this technology.

It is to be noted, however, that an automatic increase in the security and confidentiality of the applications performed on a TCP does not accompany with their deployment. This is the responsibility of the applications performed and is therefore a software issue. The undecidability theorem attests this among other things. This theorem attests that there are no generally accepted methods which enable the detection of undesired codes in software (Thompson, 2003). The assertion is based on the mappability of the named problem onto the undecidable halting problem[5]. This also applies to Trusted Computing platforms. Assertions about the behaviour of a piece of software are possible, if there has been no impact on the system at any point in time through unknown influences and applications.

5.3.1 General Shortcomings for Using TCP on the Service Side

Some of the general problems of Trusted Computing are already mentioned in section . The FIDIS Deliverable 3.9 "Study on the Impact of Trusted Computing on Identity and Identity Management" (Alkassar and Husseiki, 2007) explains in even more details regarding problems and controversial issues of Trusted Computing in general. Therefore here we concentrate on problems and shortcomings which are more specific to the usage of Trusted Computing on the service side.

A problem (which refers back to the general problems and shortcomings of Trusted Computing) is that Trusted Computing for enhancing the security / privacy of business processes on the service side might force users to eventually use Trusted Computing even on the client side implying that they also get all the drawbacks arising from Trusted Computing (e.g. threats to privacy due to the identifying endorsement key, loss of control etc.).

One of the most fundamental problems with using the current version of Trusted Computing (as specified and developed by the Trusted Computing Group) on the service side is that the specifications were made with an attacker model in mind which offers protection only against "software based" attacks—but not against "hardware based" attacks i.e. if the attacker has physical access to the hardware of the system. This attacker model might be appropriated for PCs of end users—but it is not appropriated in cases where service providers outsource some of their business processes to third parties and try to secure this outsourcing by means of Trusted Computing. In these scenarios clearly the party one wants to protect against has full access to the hardware and can manipulate them in a way which will thwart the security assumptions of Trusted Computing.

Another problem is that drawing a clear line what Trusted Computing can achieve is difficult. If we assume that Trusted Computing is absolutely perfect, why do we need any controlling (i.e. logging and auditing)? An argument is that verifying complex software with respect to trustworthiness is impossible or at least very difficult. This is especially true, if parts of the software which violates the privacy / security policy are written in a "trustworthy way" i.e. if

[5] The halting problem attests that it is not possible to calculate the scheduling of a Turing problem under all possible inputs.

the violation is not obvious. This directly leads to the conclusion that the controlling part of the software (audit and logging) and the core business logic have to be implemented by different manufactures. Moreover these parts should be certified separately by independent third parties. Fulfilling both requirements seems to be infeasible given today's market situation.

If Trusted Computing is not 100% secure (the highly realistic view) then not only the core business logic and functionality might be compromised by an attacker—but also the measures for logging and auditing might fail e.g. be manipulated by the attacker. As all of the three general principles mentioned above are based on the same assumptions about Trusted Computing they will all fail if the assumptions do not longer hold. But people might still trust the systems as they are based on "Trusted Computing", which might let them think, that the technology in itself is highly secure and trustworthy. More general: the people might tend to become less sensitive for privacy / security threats and problems, because they believe (assume), that everything now is safe due to the use of Trusted Computing.

Most of the technologies related to Trusted Computing are not widely used or even available today. The specification for usage of Trusted Computing on servers as published by the Trusted Computing group seems to be very preliminary. It does not address problems related to Trusted Computing and modern server side technologies like virtualisation and partitioning. The last revision (0.8) is from March 23rd, 2005. No visible changes seem to happen since then. Moreover many of the proposed security and privacy solutions are based on very clear and *static* overall processing environments. In practice a much more dynamic environment (e.g. new version of or patches for the operating system and the business software etc.) will be used. It is unclear at the moment if and how the integrity measurements (including remote attestation) should work there.

5.3.2 Time Problem

The security vulnerability arises through the time division of both events of the *platform attestation*, which presents a log (Mercuri, 2003) and of the *service usage*, which for security reasons can only take place after an evaluation or audit of this log. The time necessary for the evaluation of the attestation is dependent upon how much information can still be procured by the user for successful evaluation. Even if all the necessary information is available in the equipment used by the user due to earlier local interactions with this service provider, this time span will become smaller but always be present. In detail, the evaluation can comprise the examination of the following points:

- **Verification of the authenticity of the platform configuration:** With a signature of the configuration generated by an AIK, the validity of the AIK certificate is to be verified. For this, an enquiry is to be directed to the issuer, who gives information about whether the certificate has been meanwhile marked as revoked. If it is a question of an attestation by means of DAA, i.e. the platform configuration is signed with a TPM-generated key pair, this source is to be verified by DAA. For this, no further query of a third party is required.
- **Verification of the platform configuration:** In this step, it is to be verified that the service platform was not influenced by applications or components which change the properties of the service application. This task can be very complex and require the

employment of a *Property Based Attestation Proxy* (Poritz, Schunter, Herreweghen and Waidner, 2004) for mapping. For this task, additional processing time accumulates which lies between a platform attestation and the service utilization.

Figure 12 illustrates the fact that the attestation can already be outdated during the transmission to the user. The evaluation requires additional time. If this runs successfully, it does not relate to the point in time of the completed evaluation, but to the point in time of the last raised entries of the platform attestation.

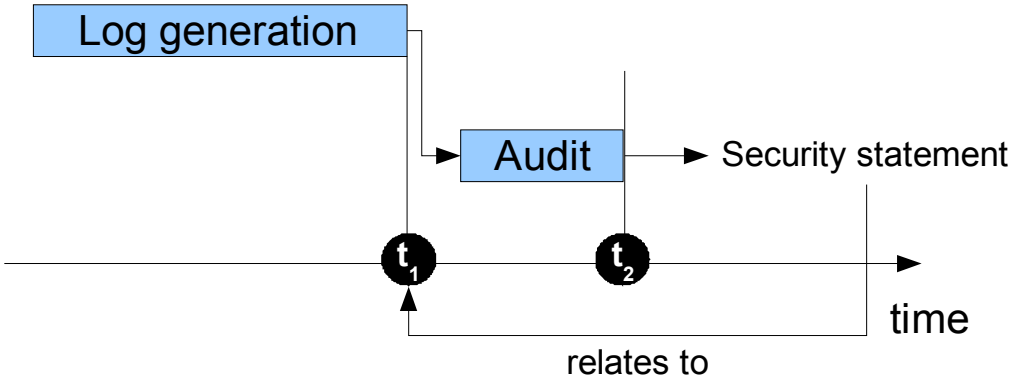

Figure 12 Log and audit collection.

From this, it can be derived that a service user uses a platform whose current state remains unverified. A service user could merely maintain that the platform was in a certain state at an earlier point in time. As this state must no longer prevail, all assertions about the system at the point in time of the attestation lose their validity due to lacking verifiability. Despite the possibility of authenticating service applications of the service-providing system and drawing conclusions from this about the behaviour of the service system, the user no longer has a secure assertion about the confidentiality at the point in time of service utilization.

This security vulnerability can be exploited by two attacks, as illustrated in the following section.

5.3.2.1 Attacks

Personal data that is to be transmitted during service utilization can be intercepted by an attacker and used elsewhere. The user would be unable to enforce his security preferences particularly when he wishes to transmit sensitive data such as biometric attributes to a third trustworthy service but an attacker manages to divert this to untrustworthy services. By an attacker trying to use system states that are consistent with the state expected by the user in order to divert the service utilization to his own unverified service applications after successful verification of the platform, he gains access to data of the service user.

This attack can take place both *internally*, i.e. within the platform as well as *externally*, i.e. from outside the platform.

Internal Attack

An internal attacker can, for example, use the time for the verification of the platform through a service user to replace ongoing service applications with his own applications. For this, it is

not absolutely necessary for an attacker to have special rights in the system. By using this same service, the attacker could terminate it in order to replace it with his own service. This attack could lead to a service user utilizing a service which was not the object of the security check performed. Another service application could have collected personal-related data of the user.

The concept used in many operating systems, so-called *privileged service access point*s[6] restricts the registration of services for users of the system with insufficient rights. This concept does not however present a satisfactory solution.

Attacks on the operating system by internal attackers are not taken into consideration here. It is assumed that the operating system used isolates users and their data and processes from one another. An isolation of ongoing processes and their storage areas safeguards against attacks on data and application code during operating time. Furthermore, it is the basic prerequisite in order that the checksum calculation of an application, as the TCG specification states, can be associated with its execution. A checksum calculation for the verification of execution would be senseless if this application could be executed in a modified way after its measurement. Present-day operating systems do not yet generally have effective isolation of memory. Approaches expected in the future that are based on micro kernels and on virtualization on the hardware side promise improvements within this context.

External Attack

An attack through an externally operating attacker constitutes a classical *man-in-the-middle attack*[7]. The external attacker thereby uses the attestation communicated by a service system to his favour by diverting the communication to his own system after successful verification. Figure 13 shows a *man-in-the-middle attack* and the point in time from which the communication can be taken over by the attacker. An attacker who has the possibility of placing himself in the communication range between the service user and a service platform which is to be attested can abuse an attestation of an external platform for his own purposes. After the successful communication of the service platform attestation to the service user (steps 1 and 2), the attacker brings the original service system to a halt in a wireless network through a flooding attack (denial of service) (step 3). The attacker now takes on the role of the original service provider with his own service applications (step 4).

While an attacker requires access to the network infrastructure for redirecting data in cable-bound networks, the technical barriers are smaller in cable-free networks, e.g. *WLAN/WiFi*. Through a *denial of service attack*[8], the connectivity of the service system can be disrupted and an attacker can direct the entire communication to himself through an address conversion.

[6] E.g. the usual privileged TCP-/UDP-Ports < 1023 of the network sockets in Unix operating systems.

[7] Attacker on the communication channel unobserved by the real communication partners.

[8] Attack through binding resources which are no longer available for the planned operation.

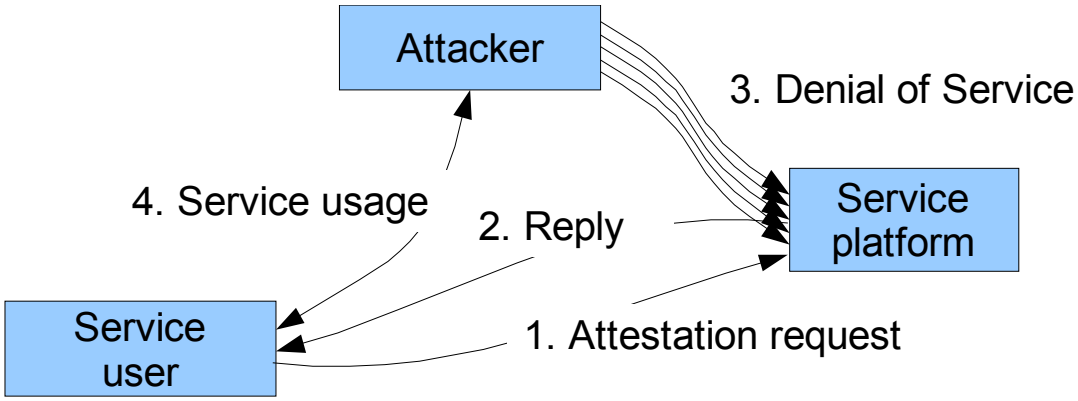

Figure 13 External attack: man-in-the-middle

It is thus possible for a service user to utilize a service which was not the object of the security verification performed. This means that it cannot be guaranteed that the user contacts his intended service and that the confidentiality of his data is endangered through release to an unintended service.

5.3.2.2 Time Problem in Detail

In the following section, the time problem and its effects on the validity of the attestation of a platform is examined in detail. For a clear presentation, the problem is divided into two perspectives. On the one hand, this is the view of the platform configuration of the service system itself; on the other hand, this is the view of the platform of the service system produced by an attestation of an enquirer. The view of the enquirer is always behind the current platform configuration. The externally visible service access points (SAP) also form part of this perspective. Figure 14 presents the view of the platform at the point in time of the attestation. The figure illustrates the state of the platform to be attested and the view of this platform from outside by a service user. The top half of the figure shows the current state of the ongoing processes and applications on this platform as well as the number of processes no longer existent (currently empty). In the table column of the view from outside, the invocation history of the applications and software components is presented in shortened form, whose checksums are extended in the PC registers in the TPM. The lower half of the figure illustrates the view of the service user of the platform from outside. This comprises the service access points accessible to him and the system state after an attestation of the platform.

The view of the platform therefore coincides with the view from outside. The invocation history of the platform and its attestation is presented in the form of an abbreviated list that itemizes the application names invoked. The number of ongoing processes reflects a partial amount of the applications invoked. The number of processes no longer existent merely serves as an illustration of the further process flow.

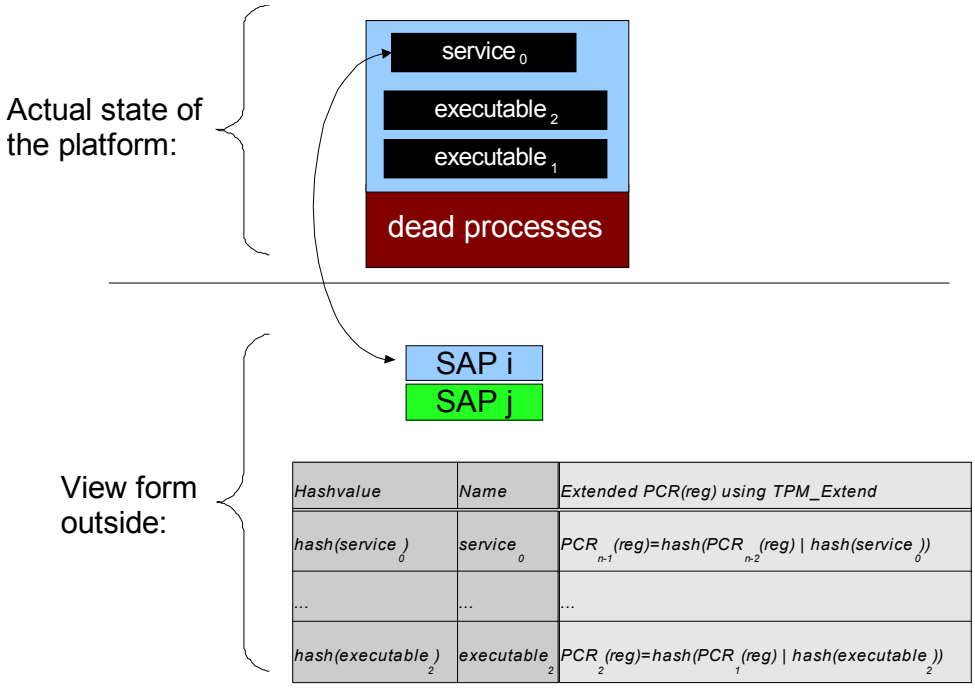

Figure 14 Platform view and view through attestation of the platform at the point in time t.

The state of the attested platform should be "frozen", which is only lifted again with service utilization. This is, however, not a good idea for several reasons. All activities of the platform, even those that are not primarily required for the execution of the desired services, must be halted. This results in

- waiting states which adversely affect the system efficiency and make it vulnerable to *denial-of-service* attacks and
- a system providing several and varied services that is unable to make any assurance about the quality of the provision of service, such as guaranteed maximum response times.

In the following, a possible development of the platform is illustrated, which can be produced by an internal attacker. The illustrated modification of the platform results in a service user contacting a service access point which was formerly registered by another service and thereby starts transferring personal data to an unintended service. If this platform modification is carried out deliberately, this can constitute an attack on the confidentiality of user data. Figure 15 schematically illustrates this modification. With regard to Figure 14, the state of the platform has changed, whereas the view from outside is unchanged, but in the meantime no longer corresponds to the actual circumstances. The decision to use a service on the part of the service user is therefore based on a no longer valid context. The inconsistency of the platform and the user perspectives can be exploited for an attack, whereby an internal attacker registers unverified applications (*malexec*) with the service access point of verified service applications (*service_0*).

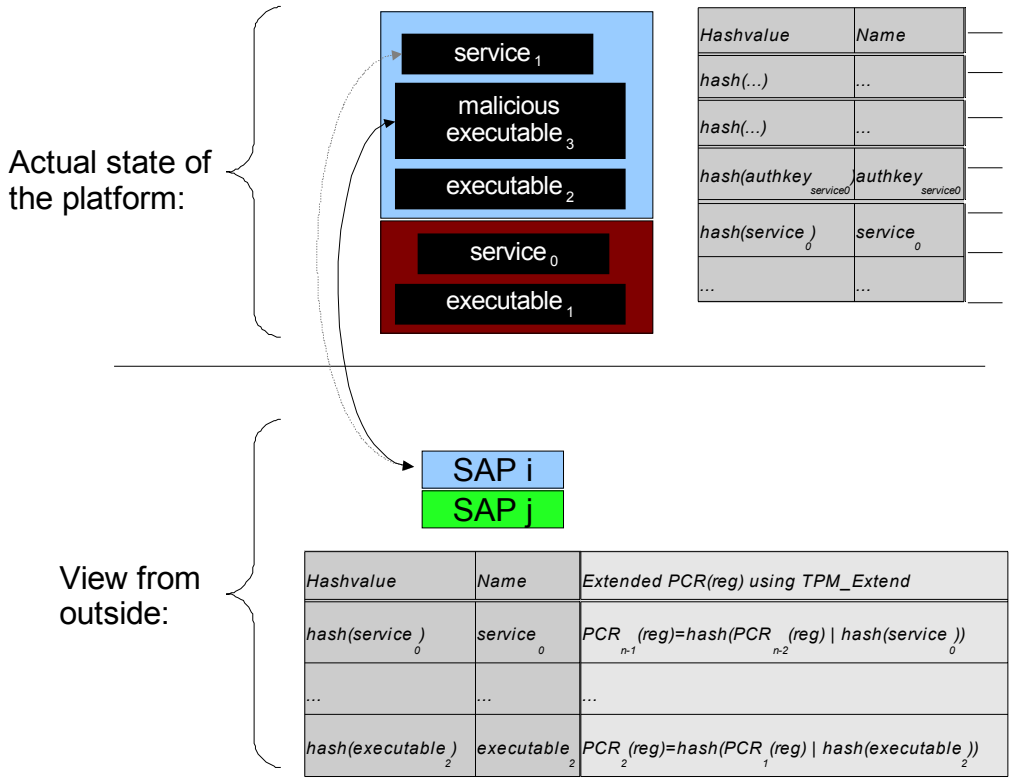

Figure 15 Platform view at the point in time t+t' and view from outside (by attestation) at point in time t.

An internal attacker can try to bring service applications to an end for example (shown in Figure 15 by the set of dead processes), so that the opportunity exists of registering the temporarily released service access point through his own service application.

The information required for the security evaluation at the point in time of service utilization shows the entries that were added after the platform attestation and therefore did not flow into the security verification. The possible difference between the platform view and the view of the enquirer is presented in Figure 16 via a PC register of the TPM. This problem basically concerns every PC register. Through the possibility of loading and executing applications, the platform changes continually. The view from outside can only be extended by a renewed attestation. This is done by means of a selected PC register which records the hash values of executed applications. Since an allocation is to be made, only a few registers are affected by a change through the start of an application.

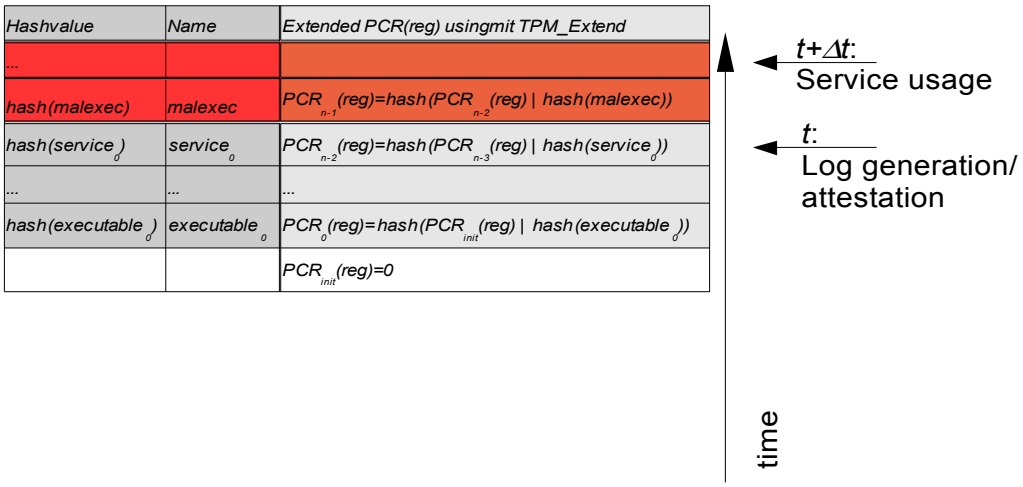

Figure 16 Difference between the two views.

The security verification of the platform based on the platform'a attestation is therefore no longer applicable. At the same time, not every modification of the platform causes the platform to be no longer in a trustworthy state. This should be made clear so that an external enquirer can recognize such modifications without any doubt and evaluate them.

5.3.3 Usability Aspects

Trusted computing may be used to implement privacy-preserving applications in a straightforward manner by deploying the respective application in a trusted environment, i.e. operating on a TCG platform. This approach is problematic mainly with regard to the evaluation of the platform configuration as part of the attestation process: The user (or an entity acting on his behalf) has to evaluate different platform configurations if different applications are used, and additional platform configurations whenever an application is patched or updated, resulting in a rather inflexible attestation process.

As a result, users may decide to restrict themselves to using only a small part of available applications based on Trusted Computing in order to avoid the overhead of additional tasks related to the evaluation of platform configurations. One possible way to improve the usability of applications based on Trusted Computing is to attest not the application itself, but instead the runtime environment in which the application is deployed. In this case, updating the application or using different applications is less problematic as long as the runtime environment configuration remains the same and does not have to be re-evaluated.

It should be noted that if the runtime environment is attested instead of the application itself, the privacy-preserving functionality has to be implemented at least implicitly via the runtime environment. A possible approach, described in detail in section 6, prevents the propagation of private data outside the runtime environment.

The next chapter presents solutions for the time problem and for verifying the usage of personal data based on Trusted Computing.

6 Approaches for Using TCP as a Foundation for Policy-compliant Data Processing

This chapter presents approaches for solving the time problem and using Trusted Computing according to the TCG specification in order to verify the behaviour of an information system by a monitor. Section 6.1 introduces the linking of an attestation and service access points at the point in time of the attestation in order to solve the time problem. By this approach, the chain of trust is given for the successive approaches for monitoring service applications and for solving the problem shown in the use case "Information Filtering". Section 6.2 presents a solution for a verifiable processing of personal data according to the requirements described in section 3.5. Section 6.2 investigates on two stages for monitoring non-certified service applications. Firstly, a service application is executed within a trusted environment (sandbox). Secondly, the data which is going to be used by the encapsulated application is divided into non-protection-worthy and protection-worthy data in order to identify those information flows which has to be considered by the user. The solution presented in section 6.3 refers to the use case "Information Filtering". This section shows the deployment of Trusted Computing in order to process private data in a confidential manner. That means that Trusted Computing is used in order to implement a trusted third party (TTP).

6.1 Solving the Time Problem by TCG attestated Service Access Points

The authenticated service access points are an answer to the danger of an attack which is present due to time differences between the platform attestation and the service utilization. The essential difference from a usual identity-based authentication of a service provider here is that a formerly attested service application is authenticated before its utilization. The identity and also other attributes of a service provider can of course be authenticated, which can additionally contribute to the increase in accountability.

The solution of the time problem is necessary as the attestation of the service platform does not allow any conclusions about the service application to be used (cf. section 5.3.2). The approach to the solution of the time problem prevents the premature obsolescence of the entire platform attestation and ensures at least its partial correctness. In addition, the introduction of a service-dependent association between the attestation and individual services leads to an enquirer being able to ascertain whether he is communicating with the desired and previously attested service or has unexpectedly got into contact with another service application. This preserves the service user from both internal and external attacks and thereby ensures that he can safely contact a service application which, in his view, is classified as trustworthy.

In section 5.2, it was illustrated that the attestation of a platform differs from the view communicated to the enquirer in the execution of further applications or in a downloading of libraries and in this form therefore no longer constitutes the suitable basis for a verification of security.

Not all entries lose their validity however when examining the attestation at process level. Assuming that the underlying operating system effectively isolates processes from one

another and thereby ensures their persistence, applications started at later points in time do not have any influence on already existing processes. Under this condition, individual attestation entries are valid up to the end of the corresponding processes, although the attestation no longer reflects the current platform state.

It is thus necessary to identify to external enquirers the entries, whose validity remains through the start of further service applications and produce a verifiable allocation in the attestation entries between the service access points and the pertaining applications.

6.1.1 Linking of Attestation and Service Access Points

In this section, a linking between the attestation of a platform and its service access points is described. This enables the user to determine whether he has contacted a service which was previously part of the attestation and therefore the security verification. Internal and external attacks can thus be detected on the part of the user and consequences drawn.

A verifiable allocation of service access points and attestation entries can be realized by a cryptographic key pair which produces a verifiable linking between the view of the service access points and the attestation. This key pair forms the basis for the service user being able to authenticate the intended service after an evaluation of the attestation. To facilitate that only the original service can successfully authenticate itself to the service user and services of external as well as internal attackers are not in a position to do so, the following properties must be fulfilled for the authentication.

- **Uniqueness:** It is to be ensured that authentication secrets are unique for the duration of a service application. A clear allocation of service access point and associated process would not be possible if the secret were to be used for the authentication of several services.
- **Unpredictability:** The schema for the selection of authentication secrets may not generate, or generate only extremely difficultly predictable key pairs for authentication. The use of the generator implemented on the TPM can be used instead of a software-based implementation for the generation of random numbers and key pairs.
- **Authentic transport:** The generated authentication secrets must be authentically and integer communicated to a user. This means that an attack on the transmission, such as a modification for instance, may not be possible or must be recognized without any doubt.

The requirement of uniqueness and unpredictability of a key pair is required by many cryptographic protocols and is implemented by a careful disposal of used key material and by scientifically discussed algorithms for key generation.

The current methods for the authentication of a service system or its transmitted data are based on certification authorities and signed server certificates. An authentic and integer transport of the key material could therefore be authentically and integer communicated in this case too.

The employment of an additional certification authority should however be abstained from to avoid further complexity. Instead of this, the required security goals can be mapped with the

trust relations already existing, which are available for the operation of trusted computer platforms.

6.1.2 Authentic Transport through Platform Attestation

It is to be definitely ensured that a user is able to authenticate the attested service applications of a platform. The authentication of the platform is thereby initially second place. Platforms complying with the TCG specification can authentically convey the state or the configuration externally by way of the remote attestation and thereby have an authentic transport channel. This channel is however restricted in the way it can only authentically transport contents of the PC registers.

These must therefore, if data, as keys for authentication, is to be authentically communicated in this way, go via the same way as applications to be executed or configurations whose checksums are filed in the PC registers. Via this loop way, the checksums of keys for the authentication of service access points can be published analogous to the checksum of applications[9].

For this, a public/private key serves as authentication key. The relationship of this key pair to precisely this platform becomes verifiable for a user through the publication of the checksum of the pertaining public key. The user receives the public part of the key at the start of the service utilization and can examine whether the service platform can correctly answer an encrypted query.

Instead of the integrity measurement of executable applications as described in the TCG specification 1.1, the checksum of the authentication key is extended into a PC register of the TPM. The TCG specification in Version 1.2 provides for more flexible handling of the PC registers:

> *The decision of whether a PCR contains a standard measurement or if the PCR is available for general use is deferred to the platform-specific specification (Trusted Computing Group, 2003b).*

The utilization of the PC registers for the checksums of authentication keys does not affect the utilization of the registers in their original function of storing application checksums. Through a clear identification marking and differentiation of keys and applications, the list can be successively validated and tested for consistency with loaded applications.

6.1.3 Linking of Service and Authentication Keys

An allocation is to be made between the checksums of executed applications contained in a platform attestation and the checksums of generated authentication keys. An authentication key can thus be allocated to each application that serves a service access point. Only if a clear allocation is given a service user can determine the checksum pertaining to the authentication key on the basis of the service intended and verify this at the start of the service utilization. As the TP module is not able to do this due to functional limitation, this lies in the task and trust area of the operating system. Two variants are conceivable as to how this allocation can be made:

[9] Via *TPM_Extend*.

- **Explicit Linking:** Through an explicit naming of the authentication key pertaining to an attested application, these can be brought into contact. It is to be noted, however, that the presentation of this linking must be clear and signed. As the platform attestation only signs the data stored in the PC registers, but the application names recorded in the invocation history by the operating system are not signed by the TPM, a further key would be required. This should however be avoided in favour of a more simple solution. Figure 17 shows the chain of checksums signed by the TP module through the attestation and its application names in the centre column of the table, which are to be signed by a further key. In addition, a signature of the program names should not imply correct names of applications and applications[10].

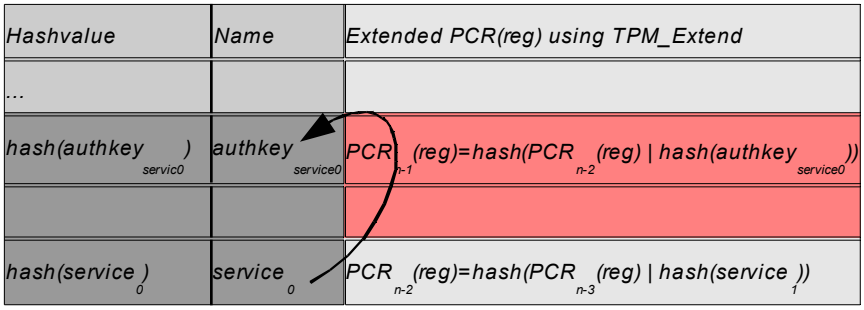

Figure 17 Explicit linking of the attestation of a process and its authentication key.

- **Implicit Linking:** An explicit naming of an authentication key can be dispensed with if its allocation to the application is clear without additional information. Thus, both the explicit linking and the necessity for its signature are inapplicable. An implicit linking can be expressed by immediate succession of the authentication key onto the application in the invocation history. Implicit linking is shown in Figure 18. The entry of the pertaining authentication key directly follows the entry of an2 application in the invocation history. A signature of the names of the hashed applications and objects can therefore be dispensed with.

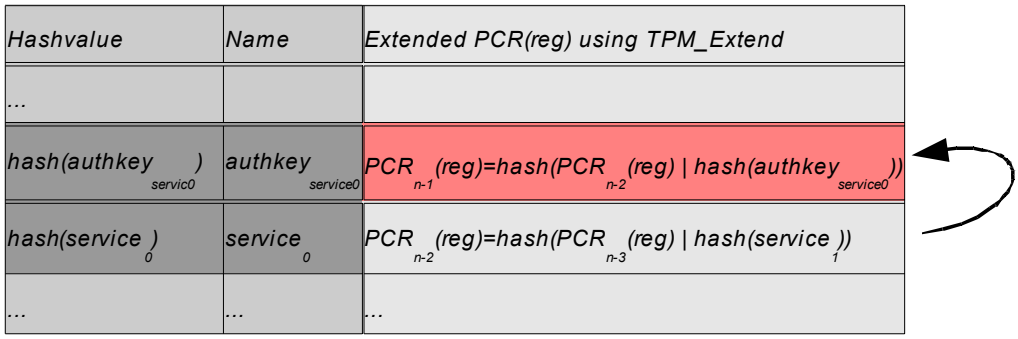

Figure 18 Implicit linking of the attestation of a process and its authentication key.

As no additional key material and thus no further trust relationship is required for implicit linking, this approach is pursued in the following for reasons of simplicity.

[10] It is quite possible that the same applications have different names or that malicious applications are concealed behind familiar system services.

The requirements of uniqueness and unpredictability of an authentication secret are achieved through the use of suitable key generators. The requirement of authentic transport is achieved by the loop way of communicating the checksums of the authentication keys authentically via the attestation. Through an attestation, the actual state of the register is signed and therewith implicitly also the chain of the checksums, which have led to this state[11].

It is only disadvantageous here that an authentication key pair must be generated for each application, irrespective of whether this registers a service access point.

Linking permits a user the authentication of the desired service application for the operating period. Owing to the process isolation anyway required for TC platforms, the start of further applications and hence the modification of the platform has no effects on an ongoing application. The user ensures by way of the authentication that it is a question of a service application which has flowed into the platform attestation and thereby also into a connected security verification. External and internal attackers who confront a user with a service application other than the original cannot be successfully authenticated by the user and are thereby recognizable as untrustworthy.

This means that a user can only successfully authenticate services which have also flowed into the platform attestation and have not been terminated. A further reason for an unsuccessful authentication can therefore be, in addition to the external and internal attacks mentioned, a terminated and restarted service with newly generated key material. This case can be counteracted with by a repeated attestation which then has to bear the checksum of the authentication key pertaining to this service.

6.1.4 Integration of Authentication in the Service Session

The service is to be authenticated at the start of the service session. For this, the public part of the key, whose checksum has in fact flowed into the attestation but has not yet been published, must be conveyed to the service user. The service then counts as being authenticated when the checksums coincide and the platform can correctly reproduce an encrypted challenge of the user. The actual service utilization subsequently follows in this session.

It is therefore necessary to integrate the authentication to be performed into the service or its service session. In order to keep the investment into the implementation of the service authentication low, it is desirable for the following requirements to be fulfilled:

- **Protocol-independency:** The implementation should provide authentication to be used by the service irrespective of the protocols applied (i.e. does not necessitate a modification of the protocols).
- **Independency of various authentication mechanisms:** An implementation should enable service-specific authentication mechanisms, provided that these do not interact.
- **Support of existing implementations:** An implementation should enable already existing service application implementations to be able to be used via authenticated service access points without being modified.

[11] Checksum functions have the character of a one-way function. It is therefore considered impossible to construct the input parameters of a given test value.

Network security protocols which are based on the session layer for the configuration of encrypted channels are suitable for this. They can enable the authentication of the session participants. It is therefore merely to be ensured that the generated key material for the authentication is used in the security protocol applied and that the utilization of such a protocol is essential when registering a service access point. Figure 19 illustrates at which point in the area of responsibility of the operating system the application is possible. The Transport Layer Security (TLS) (Dierks and Allen, 1999) and Secure SHell Protocol (SSH) (Ylonen, 1996) are suitable network security protocols. There is no need for their modification in the registration and utilization of service access points for the service application.

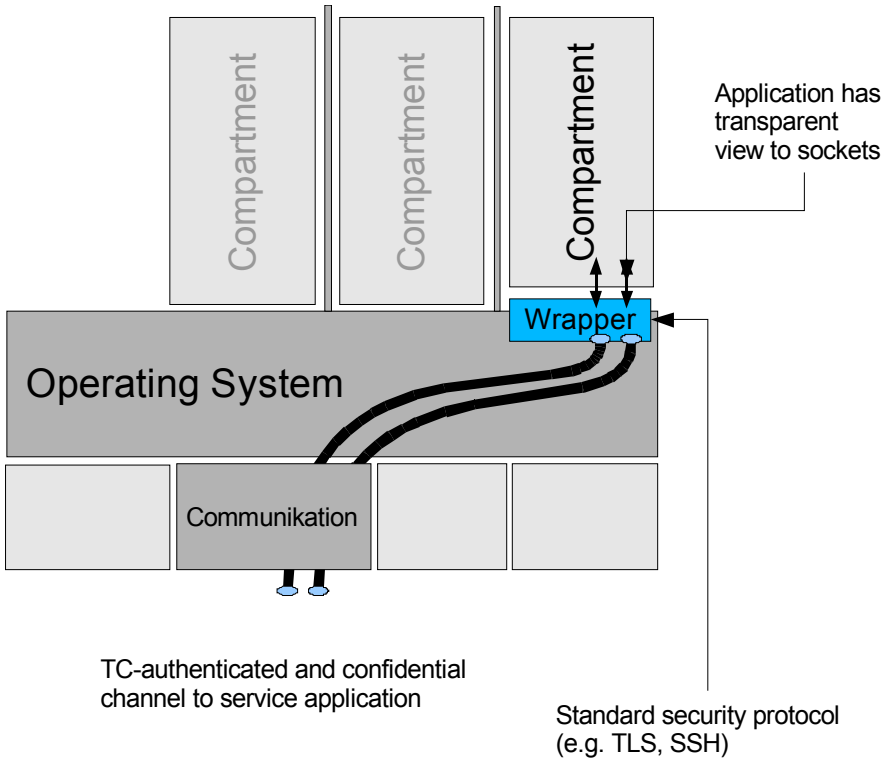

Figure 19 Application of a network security protocol through the operating system.

In summary, two individual steps are established that are necessary up to the registration of a service access point of an application. These are:

1. The start of the service application on the service platform is subdivided into:
    a) The operating system calculates the checksum of the application. It initiates an extension of the PC register intended for applications in the TPM and supplements the invocation history with new entries. It starts the application in an isolated execution environment.
    b) The operating system generates a key pair for the service application for authentication. The checksum of the public part is likewise extended in the TPM and added analogous to the invocation history. This key pair is made available for the security protocol applied. As an implicit allocation of key to the ap-

plication is used, the succession of the PC register extensions in the TPM must be guaranteed.

c) The service application registers service access points. The operating system prompts the service access points to be secured in the security protocol through tunnelling.

2. The platform is available for use with service applications.

a) The user requests the attestation of the platform.
b) The user analyses the attestation information and conducts a security verification of the platform or initiates this.

3. A user decides for or against utilization of the platform.

4. If the user has decided for utilization, he authenticates the intended application via the service access points and ensures that he has contacted the desired and attested service.

Attacks on the confidentiality when collecting data through internal or external attackers can be recognized by a failed authentication. For this, the mechanisms of the trusted computer platform according to the TCG specification form a supporting element.

6.1.5 Discussion of the Solution Approach at Issue

In addition to the implementation presented, further approaches for avoiding successful attacks when collecting data are conceivable, which are compared in the following. Some appear to offer a solution, but on closer examination prove to be inadequate or complex to realize.

6.1.5.1 Authentic notification of service access points

In order to not unintentionally connect with an undesired service, it is not adequate for an attested platform to disclose the access points in an authentic way, as this is no guarantee that this state still prevails at the point in time of service utilization and that the desired services are still at the original place. The following attacks are possible:

- An internal attacker could replace the original service application in the meantime with a defective application and register at the same service access point.
- An external attacker could divert the service utilization to another system by a *man-in-the-middle attack*.

6.1.5.2 Re-Attestation

A renewed attestation of the platform after service utilization allows limited conclusions about an orderly application flow. This solution can, however, only take effect afterwards and, furthermore, does not provide any clear indication whether an attack has taken place during the collection of personal data.

A subsequent security check in the form of a renewed attestation provides the enquirer with the history of the accessed service applications. In order to enable an enquirer to conclude from this that no attack has taken place, the following possibilities must be given:

- **All applications executed by the platform are to be verified:** Due to an unclear producible allocation between the session of the user with his service application or the remaining applications, no applications should have been on the platform that allows data collection beyond the intended service application. As the assurance of individual attributes of an application is complex, this complex process would additionally have to extend to all applications. If more than two similar service applications are executed on the same service platform, whereby the user only wishes to start a session with one selected application, applications are available on the service platform that enable a collection of data beyond the intended application. A subsequent verification is therefore not possible in every case.
- **The identification of the service platform must be possible:** As the service platform does not generate any authentication parameter, an authentication of the platform on the basis of an identity, which is based on a certificate[12] for example, must be substantiated. The authentication thereby serves to avoid a man-in-the-middle attack.

6.1.5.3 Sealing

The TCG specification describes the so-called sealing functionality. This functionality enables the binding of data to a platform and its state. This means that formerly sealed data can only be used by the platform concerned and at the same time is under a predetermined configuration.

With this functionality, a secret (secret key with pertaining certificate) on the authentication of the platform can only be made accessible[13] if this is in a predetermined trustworthy state.

A disadvantage of this solution is that the platform configurations, under which access to secrets on authentication takes place, must be known in advance and that this configuration is preserved during operation. The start of further applications however changes a platform configuration and thereby has an impact on the accessibility of a sealed secret. Since not all PC registers must compulsorily serve as basis for an access control decision about a *sealed* object, more objects could be available on a narrowly limited scale for a longer period of time. A recording of further platform modifications would then however have to be filed in PC registers, which are not used as access control criterion for sealed objects.

The sealing functionality therefore forms a suitable means of attesting the long-term state of a platform. However, it is only suitable to a limited degree for representing application-related states.

[12] A certificate can refer to the identity of the platform or its operator.

[13] Accessible does not mean the direct read access to the protected object, but merely that a signature or an encryption is possible.

6.1.5.4 Integration of the Attestation and service-specific Protocol

It would be technically conceivable to unite the attestation and the service utilization in one session and one protocol. Through this combination, the service utilization immediately follows an attestation and possible inconsistencies between the actual state and the view from outside are not present. There still remains a threat, however, if no clear allocation can be made with the service access point used or its pertaining application from an attestation within the session. An extension is therefore necessary in this case, which ensures this before utilization of the service protocols in the same session.

It is additionally disadvantageous that for each protocol an extension is to be specified which carries out a platform attestation and the authentication of the application before the actual service utilization. Existing implementations are to be accordingly adjusted by this extension on the server side as well as the client side. This is a very complex undertaking and reason for the development of so-called wrapper protocols. There are no further benefits with regard to security through the integration of attestation in service-specific protocols. The investment for implementation is, however, distinctly higher.

6.2 TCG Based Monitoring of uncertified Services

This section examines the utilization of a non-certified service application. The application was not examined by an independent entity and therefore a user has no information whether a confidential processing of protection-worthy data takes place within this application. The service application can therefore also be regarded as a so-called *black box* (see Figure 20), about the insides of which no information is obtainable. The behaviour of a service application with regard to use, storage and transmission of personal data is not transparent to the user.

If no technical measures are used for the protection of personal data, the user must trust a service application and its operator that processing only takes place within the framework of the agreement. In the following, the processing of protection-worthy data is therefore to be made transparent to the user. Even if a misuse of data thereby cannot be prevented, it can be established after service utilization whether confidential processing has taken place. Possible violations can thus be later identified and lay the foundation for further steps, e.g. legal sanctioning.

It is assumed that a service provider behaves according to the prevailing regulations and user requirements, in order to not risk any detriment through possible misconduct in the form of legal steps, a customer withdrawal or the loss of his reputation. Even if a remaining risk of misuse through a subsequent evaluation of the processing of personal data cannot be dispelled, the transparency of a processing provides the incentive for a service provider to behave according to a published policy or one prescribed by the user.

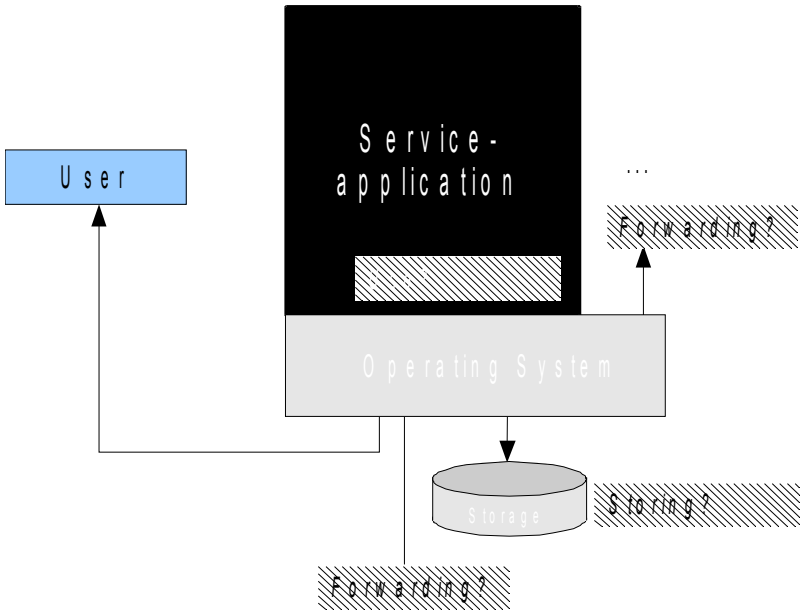

Figure 20 View of the service application as unknown application.

From the point of view of a service provider, the observation of the execution of a service application offers several advantages compared with a certification or the publishing of the source code of a service application:

- **No necessity for certification of the application:** It is not necessary to subject a service application to an intricate certification or a test.
- **Applications can be modified:** Due to the unrequired certification, a re-certification is dispensed with, which is necessary for a service application modification.
- **Flexible implementation of a data protection guideline:** It is left to the service provider as to how he enforces his data protection guideline.
- **Publication of the source code is unnecessary:** The publication of the source code and the related drawbacks can be avoided.
- **Test of the user's observance of a policy:** The service provider can use a transparent processing to examine his own service application for confidential processing. Possible violations of a data protection guideline can be hereby detected and subsequently dealt with.

As a user draws on the protocol of the execution as an indication of the confidential processing of protection-worthy data, its authenticity must be ensured. This involves the authentic generation and recording by a generally accepted component for monitoring, whose function mode is confirmed by an independent certifier. Furthermore, it involves the authentic transport of data to the user by a secure logging protocol. The authentic transport of log data is described in (Accorsi, 2005) and is not closely examined here.

The solution approach presented in the following examines the logging of the processing of protection-worthy data in a certified service application by a certified monitoring component.

This constitutes the basis for a transparent processing and evaluation, which can detect a possible misconduct of the service provider.

6.2.1 Service Application as a Black Box

It is assumed that a service application is available in binary code and does not allow any insight into the application logic so that the behaviour with regard to the application, storage and transmission can be drawn or derived. The application is therefore regarded as a *black box*.

For already existing and non-certified applications, only the observation of the behaviour during execution remains. The aim thereby is to observe and protocol (generation of a log) the application flow as precisely as possible. Possible violations with regard to the application, storage and transmission can thus be later analyzed through an audit of the generated log data.

A further requirement is the execution of the service application in a sealed environment. This protects the application itself from unauthorized read/write access and a readout or modification of its memory areas. A monitor that controls the service application and logs its events is to be especially protected from unauthorized access. Otherwise, the quality of the logging cannot be ensured.

6.2.2 Encapsulation of the Service Application

If one assumes a compiled service application, there are only limited possibilities to gain information from the binary code about the behaviour with regard to collection, storage and delegation of personal data. The remaining possibility is to let the application run in an environment that observes the execution and protocols a log. This environment, referred to in the following as encapsulation, can, however, only log events that are detectable from its point of view. In the case of a compiled application, these are system calls to the operating system for storage and transmission, but not via the use of data within the application. Figure 21 shows the encapsulation of a service application in an execution environment. A monitor between the application and the interfaces of the underlying operating system observes the communication. Outsiders, such as the user of an encapsulated service application, must thereby be able to rely on the quality of the data collection through the monitor. The monitor must log all events and prepare them in such a way that a subsequent modification or removal of individual log entries is not possible or at least detectable (Schneier and Kelsey, 1999; Accorsi, 2005).

6.2.2.1 Authentication of the Encapsulation

Encapsulation with the respective monitor forms the basis for the logging of the application execution and is responsible for the quality of the log generated. A user must therefore be able to assure himself of the presence of an encapsulation which satisfies his requirements of a protocol. The user can authenticate a certified encapsulation to that effect with a service application executed therein through the TC-authenticated service access points.

Instead of authenticating a service application, the encapsulation with the service application executed within is authenticated. The use of TC-authenticated service access points ensures the confidential delivery of protection-worthy data to the service application executed within

the encapsulation. At the same time, it gives the certainty of transferring data into an environment that provides assertions about the processing.

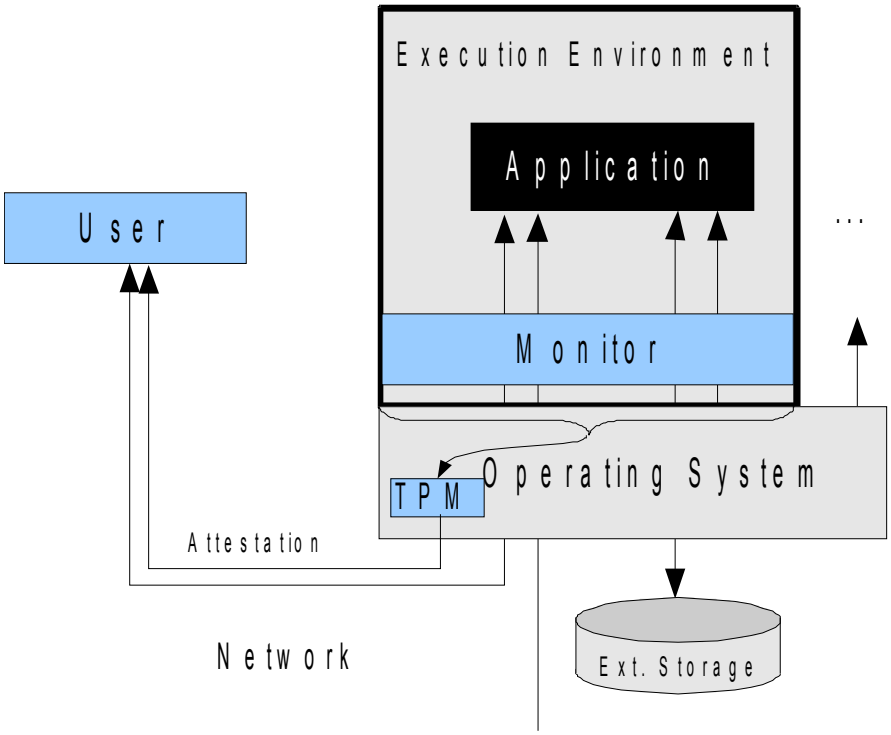

Figure 21 Encapsulation of an unknown application in an execution environment.

The Figure 21 shows an unknown application (therefore considered to be a black box application) which runs in a sealed execution environment. A monitor controls the incoming and outgoing communication of the execution environment and protocols the events in a log. In order that outside users can make an assertion about the data quality of the monitor, it is necessary to be able to authenticate this in the form of a service application. An available TPM and the TC-authenticated service access points, serve for this, which provide this evidence and allow the authentication of the execution environment when utilizing its service access points. The monitor of the execution environment must thereby control all available communication channels. The generated log can be analyzed by an audit.

6.2.2.2 Criteria for an Authenticated Encapsulation

An authenticated encapsulation must fulfil important requirements in order to make a possible anomaly noticeable when processing user data with individual user preferences. As the observance of an application by the monitor of the encapsulation cannot avoid but only log any anomalous processing, it is additionally necessary to be able to identify the service provider.

- **Identification of the service provider:** The identity of the service provider must be able to be clearly established in order to be able to bring him to account in the event of a violation. The authentication of the application alone, as with the collection and processing through a certified service application, is not adequate in this case as

violations owing to non-certified applications are possible. The execution environment and the monitor must be authentifiable however, as they are responsible for the provision of an authentic base data.

- **Identification and monitoring of the communication contents:** The encapsulation must be in a position to consistently monitor the communication led by the application and its contents. This includes the communication via networks and the protocols used thereby, so-called Pipes[14], Shared Memory[15], Semaphores[16] and the communication via files. All ways suitable for transmitting information are basically of significance. However, a monitoring inevitably comes up against boundaries as it can observe known communication channels only. So-called *Covert Channels*[17] are not thereby detectable. Service applications that were designed with regard to an execution in a sealed environment could theoretically use different variants of concealed channels in order to avoid identification and thus the monitoring of a communication.
- **Identification of the communication end points:** The encapsulation must be in a position to clearly identify the end points of a communication like source/destination service access points of a network communication and of pipes, semaphores etc.
- **Classification of confidential data channels:** The extent of the collection of data via the communication activity of the encapsulated application can depend on the communication end points and the classification of their trustworthiness. The classification of communication partners into trustworthy or untrustworthy partners thus reflects in the amount of log data and can serve to reduce data emergence. The communication channel between the encapsulated service application and an unknown communication end point is to be observed more precisely, for instance, than the communication channel between the user and the encapsulated application.

Furthermore, the encapsulation of a service application should have properties which enable a collection or an allocation of events to a service session.

- **Relating a service application to the user:** The logged events of a service application must be able to be connected with the user responsible. By this is implied that a service application may only serve one session with one service user simultaneously. Alternatively, a service application serving several sessions simultaneously could make an allocation of logged events itself. This means, however, that responsibilities and the functionality for generating a log are hereby shifted into the application. However, this endangers the independency of the monitor from the application.

The events of the encapsulated application must be authentic and be integer transmitted to the user. For this, the generation and the transport of the log must satisfy the following requirements:

[14] Communication streams between processes.

[15] Memory areas jointly used by several processes.

[16] Marks for signalizing a state.

[17] Covert channels, e.g. in the form of temporal response behaviour.

- **Generation of authentic log data:** The monitor in the encapsulation displays the events of the application carried out. It is of interest to a user that a monitor is used that implements the set requirements. One therefore only relies on the log of a monitor, to which the observance of the set requirements is confirmed or certified. The presence of an encapsulation with certified monitor can be verified with the TC-authenticated service access points presented.
- **Authentic and integer transport of the log data:** The log data collected is to be made available to the user. The authenticity and integrity of the data during transport, e.g. through previous checksum formation, signature and time stamp, is to be thereby ensured (Hohl, 2006).

6.2.2.3 Evaluation of a Service Usage Log

The monitor of an encapsulated service application records all communication and storage events. Excluded from this are non-detectable side channels which the monitor cannot detect. However, no assertion can thereby be made as to whether protection-worthy data has left the application by an unobserved communication or storage.

To verify this, application-dependent filters would be required which also have insight into the contents of the communication and detect the transmission of protection-worthy data. However, the encapsulation would lose its independency of the application. Furthermore, application-dependent filters would have to have the necessary algorithms and keys for decipherment when an encrypted communication takes place. This also poses an intervention in the encapsulated application.

As the content of a communication or memory cannot be more closely classified, it must be assumed that not only the communication between the service user and the encapsulated application, but any other communication transports and transmits protection-worthy data.

The conceptional weakness of the encapsulation of non-certified service applications bases on the fact that suspicious factors can only be dispelled by an absence of events. Observance of the information flows of protection-worthy data within the service application is not possible.

Proof through Communication not Taking Place

In the following, utilization restrictions when processing personal data are listed whose observance can be verified by means of the log generated. It is assumed that these restrictions were formulated in the form of a security guideline by the user before service utilization or an existing guideline was accepted. The specification language used must be at least able to specify the following cases:

- **No permission for transmission:** A transmission of data after the utilization of a service can be ruled out beyond any doubt if no communication to other computers or processes has taken place. Storage of data on persistent data carriers only presents a problem in this connection if data access of written data is not limited to the encapsulated application. A cryptographic binding of encapsulated application and its data can guarantee this.
- **No permission for persistent storage:** A persistent storage of data, which can include personal attributes of the user, can be detected through the absence of write accesses

to persistent data carriers. However, the case is to be considered where data was transmitted. The restriction of non-persistent storage is consequently also transmitted to sub-services. One possibility for the inclusion of sub-services is shown in Figure 22. The log of all service platforms involved may therefore not contain any entries about the storage of data. Provision for this is that:

  - trustworthy log data also is generated on the sub-services. The primary service can examine each sub-service by a platform attestation and classify the communication channels as trustworthy if the service applications of the sub-services are likewise executed in encapsulated execution environments with monitors.
  - the log data of all sub-services involved is also accessible to the user for verification.

- **No permission for transmission and storage:** The implementation of this requirement can be detected by the absence of entries on the communication and on the storage in the log of the encapsulated application on the service-providing platform. The inclusion of sub-services is excluded.

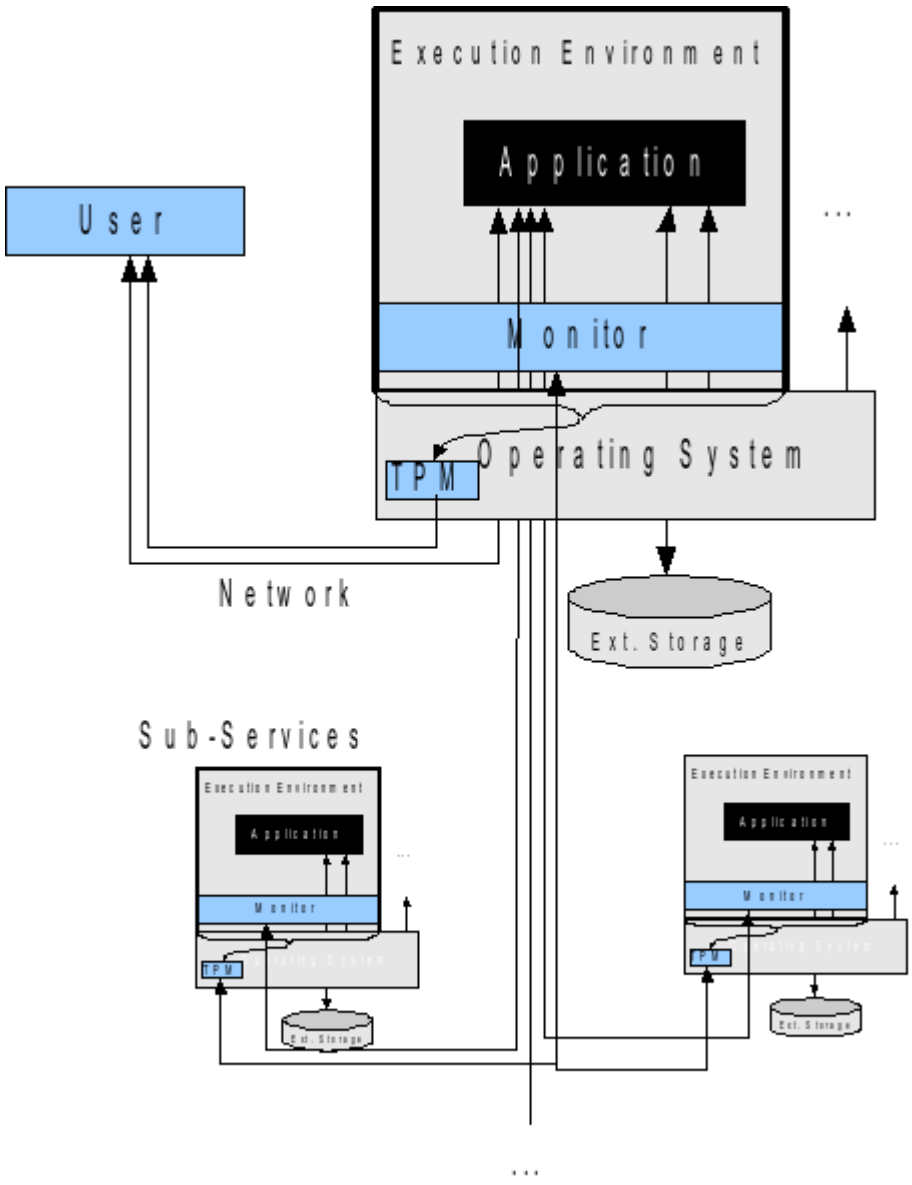

Figure 22 Service with Sub-Services.

If the service utilized by the user is based on so-called sub-services, formulated user requirements, such as no persistent storage of the data are transmitted to these services. In order that an evaluation of the logging can subsequently take place, all platforms involved must execute their services in a logging environment and the generated log must be made accessible to the user.

Uncertainty with Existent Communication

The restrictions when monitoring a communication from the standpoint of an external monitor make it difficult or impossible to make an assertion whether protection-worthy data is an

integral part of the content of a communication. A monitor could be fitted with a row of filters for standard protocols to get a detailed insight into the events. This may be possible for a few protocols. However, due to the multitude of protocols, this approach is too involved and fails even when using encryption. Moreover, no easy to filter and general format for presenting personal data is known which meets the requirements of various applications and could be used accordingly.

Definite assertions can therefore only be derived from the absence of events concerning transmission and storage. Even if services, which are dependent on the communication with other services for carrying out a job, do not communicate any personal data, a suspicious factor is roused by the pertaining log.

6.2.2.4 Model: The Oblivious Terminal

The model of an oblivious system was applied to a service in a hospital environment. In the example scenario, a patient can take a look at protection-worthy data, such as his hospital records, at a public terminal. In this example, this does not endanger the privacy of the patient, irrespective of the software used in the terminal.

In contrast to an encapsulation which only observes, a functionally limited encapsulation is used here. This prevents unwanted information flows and ensures confidential processing, irrespective of the software used. Resource restrictions and the functionality of the encapsulation are thus transmitted to the service application executed therein. Through the verification of such restrictions, properties regarding the protection of protection-worthy data are recognizable.

This was achieved by the use of an encapsulation that does not have any outgoing data channels other than those for the communication with the user, and only has limited storage capacity. The application executed therein is not in a position to store or transmit protection-worthy data over the period in time of several sessions. Instead, storage space must be again released. Obliviousness is thereby inevitably implicated. Owing to the service capacity of present-day systems that are equipped with extensive storage capacity and communication possibilities, an artificial restriction must be applied. Analogous to virtualization approaches such as VMWare[18], *chroot*[19] or *jail*[20], an encapsulation is mapped and implemented on a high-capacity system through an additional abstraction level between application and system.

A prototype of a service for the retrieval of protection-worthy data was realized by means of a Unix operating system. An attested encapsulation with limited storage and communication possibilities was thereby implemented[21] and is consequently unable to persistently store and transmit data.

The user of the terminal initiates a service session by inserting the health chip card. Before the health chip card of the patient guarantees the terminal access to the stored content, the card

[18] See http://www.vmware.com

[19] See http://www.annodex.net/cgi-bin/man2html?chroot+8

[20] See http://docs.frieebsd.org/44doc/papers/jail/jail.html

[21] Further details can be found in (Hohl and Zugenmaier, 2005).

first examines the terminal and for this purpose demands a disclosure of the state[22]. If the terminal can authenticate itself successfully towards the health chip card as a terminal which only provides the contents of the health chip card with an encapsulation with the above-mentioned restrictions, the interaction is continued. The obliviousness of the system was technically implemented by a virtual machine which has the named restrictions and after termination of the session, or on removal of the health chip card, performs a cessation of the session by overwriting the used memory area.

In Figure 23, the technical setup is shown that initiates a service application in an encapsulated environment on the Unix derivate NetBSD and makes its checksums accessible to the health chip card. The figure shows the prototypical setup of an oblivious terminal. The application of the terminal is thereby carried out within a sealed environment which cannot be overcome by internally running applications. The properties of the sealed environment are thereby transmitted to the executed application. The health card of a patient examines the configuration of the terminal and of the execution of the application in the sealed environment. The card only ensures access to stored data when this step has been successfully completed. If the terminal cannot prove the presence of a sealed environment, the communication is terminated.

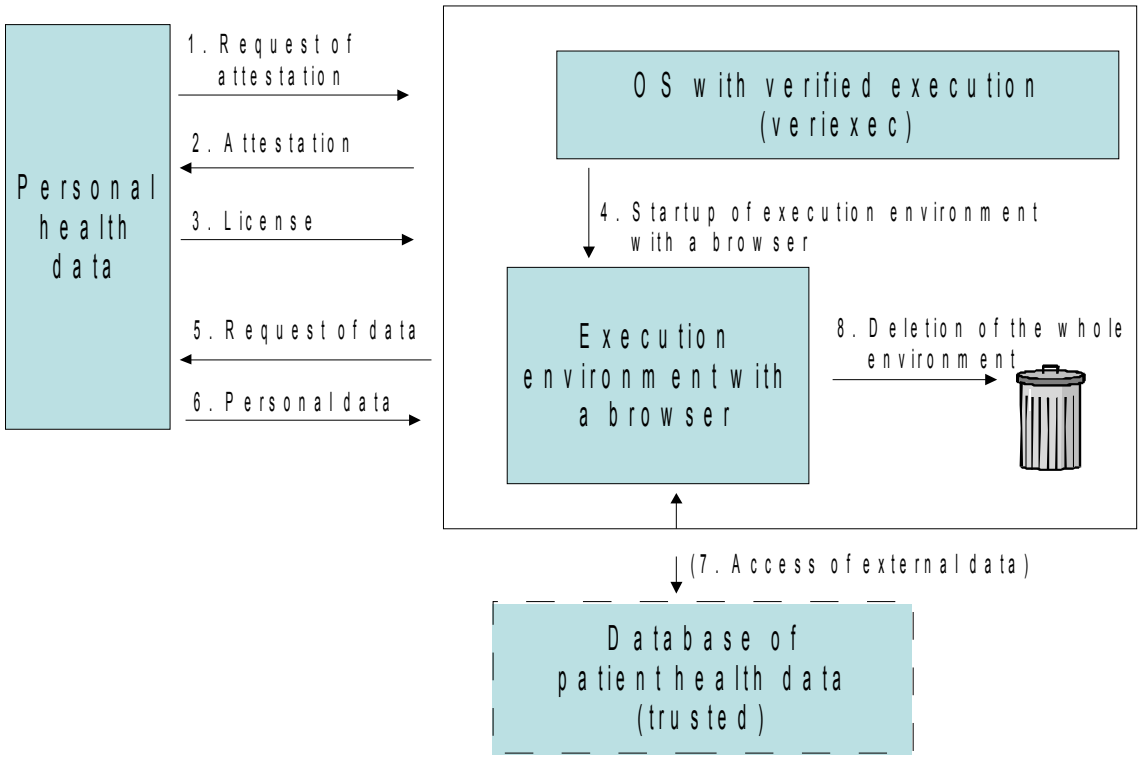

Figure 23 Prototypical realization of an oblivious terminal.

The course of the communication is shown in Figure 24 and shows the verification of the execution environment through the user's card. If the execution environment of the terminal

[22] This disclosure can be regarded as equivalent to the attestation with systems in accordance with the TCG specification (Trusted Computing Group, 2003a).

does not correspond to an expected value known to the card, access to the card is not given and the transaction is thus discontinued.

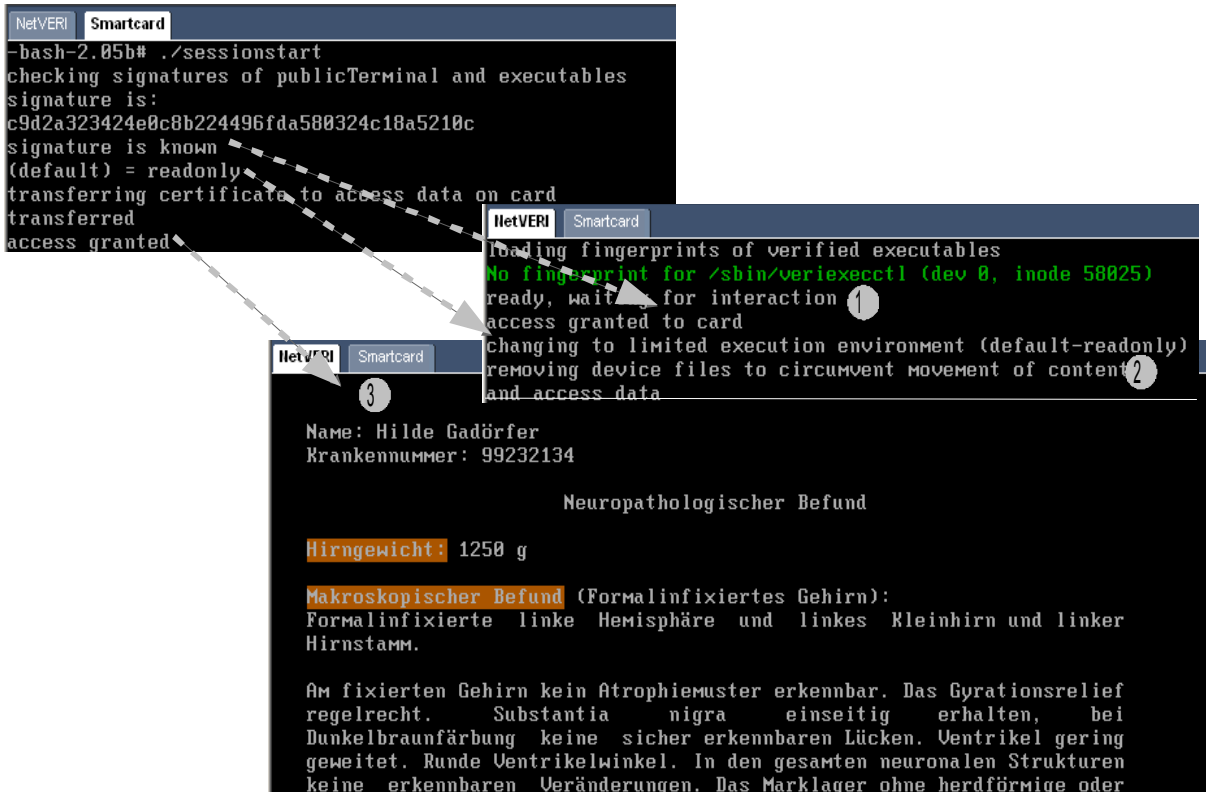

Figure 24 Course of the communication with oblivious terminal.

The figure illustrates the card and terminal outputs. The functionality of the card was thereby mapped on a standard operating system. The card examines the execution environment (step 1) of the terminal. If this or its checksum is known to the card, a read only access of the terminal to data stored on the card is allowed. In addition, the communication and storage possibilities of the execution environment are limited (step 2) and the service initiated (step 3). If the execution environment of the terminal cannot authenticate itself successfully to the card, access is denied.

Through the encapsulation restrictions which are transmitted to the executed application, a permanent storage, transmission and thereby also use of protection-worthy user data elsewhere is prevented, irrespective of the service application used. An encapsulation that identifies and logs precisely these data channels can prove to a service user whether protection-worthy data was stored or transmitted.

It is therefore adequate for a user to be able to examine in advance whether his data is collected through an encapsulated application with the aforesaid restrictions. However, a functionally restricted encapsulation cannot be used with service applications that, for service provision, have to access persistently stored data or are dependent on a communication with other services.

6.2.2.5 Conclusion

The encapsulation of an application in a sealed environment allows the observation of the communication and in this way enables the logging of the workflow. Provided that the service application is only available in the binary code, this appears to be the only possibility for a service user to gain insight into the processing.

The logged execution of a service application can be used for applications with simple communication behaviour and proves to its users that their protection-worthy data was not stored or transmitted. The encapsulation provides a tool for these applications which can improve the acceptance of this application due to the certified reporting through the monitor and analyzability.

The aim of being able to trace the processing of protection-worthy data in a way that processing deviating from a policy can be established beyond any doubt cannot be completely achieved. The log generated from the view of system invocations is too inaccurate to be able to evaluate more closely the cases where a communication has taken place and uncertainty about the contents transmitted exists. For this, approaches are necessary that allow assertions about the incoming and outgoing contents of a communication.

It would then be possible to make an assertion whether protection-worthy and indicated data has been transmitted to a communication interface or a persistent memory. On the one hand, this allows an exact logging and evaluation of a log with regard to storage and transmission for frequently communicating applications. On the other hand, the number of suspicious factors declines, as the communication becomes classifiable into protection-worthy and protection-unworthy data. The employment of the information flow analysis examines this case in the following section.

A conceptual weakness of the monitoring and logging of events is the uncertainty as to when an application has concluded the processing of a service. This point in time must not necessarily coincide with the point in time of the completion of a session and could lie temporally behind this. The evaluation of an incomplete log can therefore not detect a dangerous situation due to lacking entries. A further restriction is that during the execution of the service applications, only one service user can be served. When dealing with several users simultaneously, the clear allocation of the log of a session to the respective service user cannot be ensured. The monitor of the execution environment can detect several users merely by means of several communication relations. A subsequent allocation of further events is not possible. Service applications are therefore to be laid out in such a way that each session is operated by a separate service application.

6.2.3 Encapsulation by Information Flow Analysis

This section investigates on the employment of the so-called information flow analysis (Tse and Zdancewic, 2004) for an encapsulation of a service application for logging events. The aim is to obtain information about the contents of a communication and storage. By this, a communication or storage of non-protection-worthy data should be differentiated from protection-worthy data. The number of suspicious factors should be reduced and distinguishable from actual cases of misuse.

The technical prerequisite for this is that a security typed programming language has been used for implementing the service application. The aim is to trace the information flows in the application in order to be able to better classify communication channels.

A decisive feature of a typed language is the so-called *type inference*. This means that the type of each function or each printout can be concluded from their individual criteria without having to explicitly specify this. Irrespective of whether it involves a *static* or *dynamic* type inference, this task is derived during the transmission period of an application or during execution.

Security typed languages extend typing with security types. Academic representatives of these application languages are *FlowCAML* (Simonet, 2002) and *JIF* (Myers and Liskov, 2000). Typed languages which differentiate between the types of variables and objects, types for designating confidentiality or integrity are used for security typed languages. By means of this marking, the flow of information through the application can be analyzed via the type inference[23]. Data once provided with a type carry this during the entire application execution. The analysis of the information flow takes place on the basis of the security types. It thereby becomes apparent whether information has flowed to or from a typed date or object. The information flow can be traced on this basis and is also controllable via an arrangement of the security types. The information flow analysis thus forms an access control model for confidentiality at variable or object level.

Applied to objects in which protection-worthy data is filed, the type inference ensures that security types are continued and derived irrespective of the processing of data carried out. It is therefore necessary to mark data objects for storing confidential data at the outset by a security type suited to the confidentiality. In Figure 25, a security type must be allocated for the user data collected that can be recognized again by the monitor during a communication.

The prerequisite of this approach is a proper functioning of the type inference and the initial allocation of security types. Through the use of TCPs, it can be verified to the user whether certified software is used that implements an execution environment with type inference and an initial allocation of security types. The application does not have to be an integral part of the software components to be certified, whereby the certification outlay is restricted to one execution environment with information flow analysis and to the allocation of security types.

6.2.3.1 Solution Approach

The execution of non-certified applications likewise takes place in a sealed execution environment. However, the monitor has an information flow analysis with which the information flow of typed objects and communication interfaces can be monitored. For this, the service application must be written in an application language with support for the security types. The presence of this execution environment must be verifiable and have the following functionalities:

[23] A differentiation is made between direct data information flow, e.g. through an allocation, and indirect information flow, e.g. through an if/else case differentiation. The condition leading to the leap into a case differentiation has impacts on objects outside the case differentiation. This constitutes an indirect information flow.

- **Allocation of security labels:** The execution environment allocates security types to objects before processing to indicate their confidentiality level.
- **Type inference:** The type inference takes care of the transmission of security types to objects that are dependent on confidential data and could flow off via the personal data.
- **Monitor:** The monitor in the execution environment supervises information flowing away via communication channels by means of allocated or derived security types. Objects that are designated with a confidential security type can be regarded as confidential information flowing away.

The certified execution environment with the initial allocation of security types for indicating protection-worthy data must, in turn, be authentifiable by the user. The quality of the entries of the log produced depends on this. Certified execution environments are identified by the Remote Attestation functionality of a TCG-compliant platform. The TC-authenticated service access points serve to prevent possible attacks when transmitting protection-worthy data.

The execution environment of the encapsulated service application supports the information flow analysis. With the communication of data into the encapsulation, personal data of the user is provided with security types. The type inference ensures a continuation of the security types during processing through the application. If sensitive data is communicated by the application, this can be detected by the security label types and logged. Confidential contents are distinguishable from non-confidential contents on account of the security types and the log precision improves. The proper allocation of security types and type inference must be ensured. The use of Trusted Computing platforms serves here for identifying the components for this task.

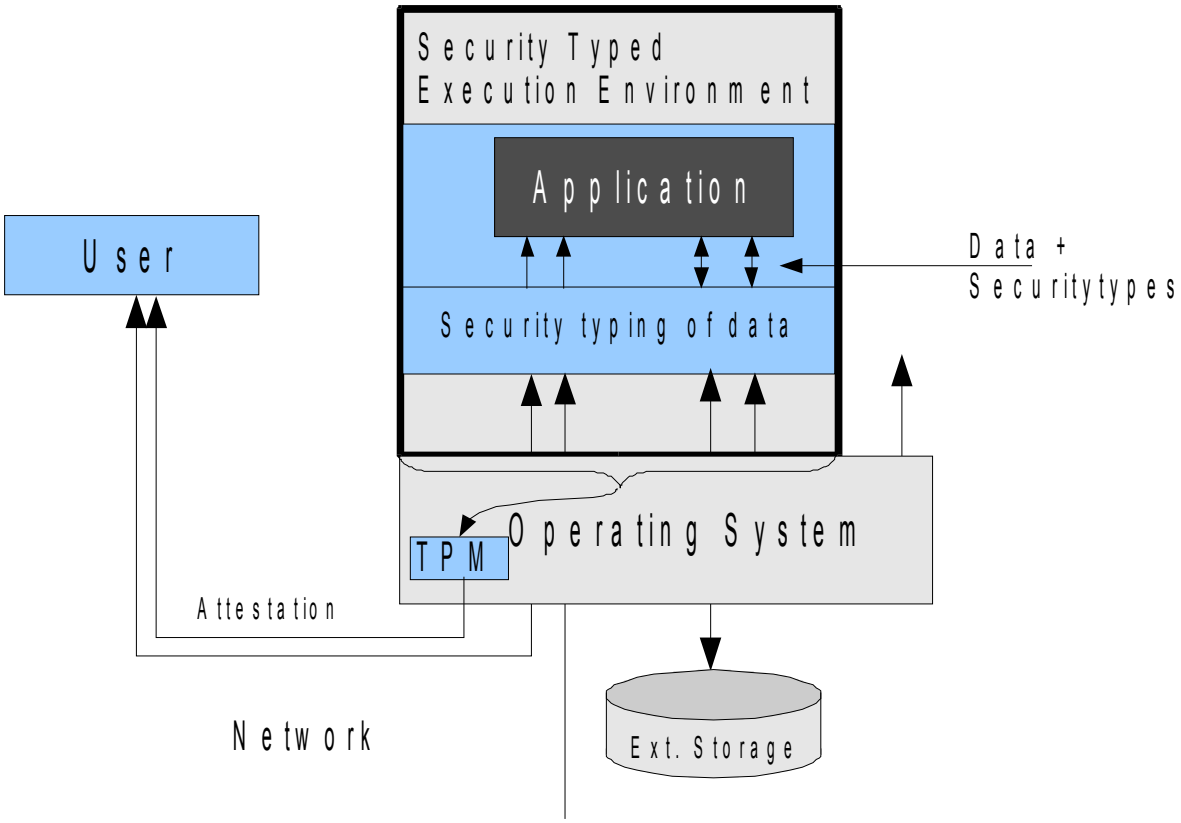

Figure 25 Encapsulation of an Application with Information Flow Analysis and Security Types.

6.2.3.2 Experiment: Tariff Calculation

On the basis of a sample application, the suitability of the information flow analysis as monitor for supervising a service application was examined. For this, a personalised service was produced which constitutes a tariff calculator for a life insurance and takes into account protection-worthy user data when generating the tariff. A user of this service is, in addition to an individual tariff generation, also interested in his transmitted attributes being confidentially processed. Furthermore, he wants to have the opportunity of acquiring knowledge as to whether data about him is being stored or transmitted.

To realize the sample application, the information flow analysis of the FlowCAML application language was used. With the aid of type inference, the security types and an access control model can be mapped on the basis of a hierarchical arrangement of the security labels. This arrangement determines the information flows permitted. It is formulated in the form of a so-called *flow* statement. With the static information flow analysis, allocations are examined at the point in time of the application compilation on the basis of the hierarchical arrangement of security types. An inadmissible allocation is then excluded during the application cycle. As the previous information analysis included all possible information flows compared to the hierarchical arrangement of the types at the point in time of the compilation, the information flow analysis was not used in the experiment at the point in time of compilation.

The information flows that took place during the execution that are caused by the processed instructions on the execution path are of interest. The interactive type inference interpreter, which derives the security types, serves to determine the actual information flows. The application is thereby executed up to an assertion where the information flow analysis determines an unauthorized allocation owing to a security type. This violation is written down in the log and the execution continued.

An allocation of security types which classify incoming data was consequently made for all input and output channels. This step is of wide significance, since the results of the service application cannot be interpreted with incomplete allocation or an allocation of the security types unsuitable to the application situation. Therefore, not only do network interfaces represent communication channels, but also the input/output of the consoles and read and write accesses to persistent data memory.

6.2.3.3 Classification of Security Types

Protection-worthy data is transmitted via the communication channel from the user to the encapsulated service application. Before processing, it is to be allocated a security type, by means of which the type inference can determine the flow of the data transmitted in the application. Since it is conceivable that protection-worthy data contains attributes which allow a direct reference (e.g. via a bank connection) or an indirect one (e.g. via gender), the transmitted data was divided into confidential and less confidential attributes. This enables the transmission of attributes not classified as confidential to sub-services without inference to the user concerned. Classification of the attributes can thereby depend on the application scenario and was organized as follows in the scenario of the sample application.

Attributes such as name, age, address, income and bank data were classified by the *confidential* security type. The attributes of marital status, occupation, gender, smoking habits and sporting activities were classed as less confidential by the security type *unclassified*. Figure 26 illustrates this.

This classification is further refined analogous to Figure 27. A separate security type is thereby allocated to each attribute in order to get a more exact assertion about the information flows that have taken place.

6.2.3.4 Logging by Information Flow Analysis

For the generation of a log of the data flows during an application, an interactive tool was used for the derivation of type inferences. The sample applications were executed stepwise therein and the log of the information flows presented in Figure 26 and Figure 27 generated. Since FlowCAML makes access control decisions on the basis of security labels and their hierarchical ranking, no ranking was defined for this experiment in the first instance. During execution of the application, operations and accesses to objects are detected that are not permitted due to a non-specified information flow. At the same time, the type inference system derives the policy necessary for this instruction. It leads to the log of the information flow that has taken place. Subsequently, the information flow which led to the violation is allowed in order to be able to carry out the instruction.

Protection-worthy personal data are classified according to their confidentiality with the security type *confidential* and the type *unclassified*. The objects of the insurance are marked with the security type *insurance* and the standard output channel with the security type *stdout*. The underlying application for the log in Figure 26 reads in some personal data of the user and by case differentiation on the basis of certain attributes calculates a tariff, which is subsequently issued. The left side shows the processing steps and the personal-related attributes involved. The log of the information flow that has taken place is illustrated in Figure 26 on the right-hand side.

Figure 26 shows the information flows from objects with the type *confidential* and the type *unclassified*, which have influenced objects of the *insurance* type and were released via the standard output channel. A more precise utilization of individual attributes of the service user is not attainable with the rough classification of user attributes into two security type categories. A more exact allocation of security labels in Figure 27 solves this problem and indicates or eliminates a flow of confidential information through the log of the output channels. This permits a differentiated view in contrast to the observation of the communication and storage of an encapsulated application without information flow analysis.

Instead of the classification of several attributes through one security type (as in Figure 26), each individual attribute was provided with its own security type. This has direct effects on the generated log, which is a transcript about the attributes used. It can now be detected, particularly by the communication and storage of the tariff, whether data is transmitted with a security type and which data it concerns.

The first step in Figure 26 shows the influence of the tariff through personal attributes of a user. Since the attribute income is evaluated with the type *vertraulich/confidential*, a flow of data of the type *vertraulich/confidential* to data of the type *versicherung/insurance* is logged. Further access to objects of the same type leads to the same output. They do not contribute to the later evaluation of the log (step 2). An alteration of the log is triggered by the evaluation of an attribute with another security type (step 3). As the output channels are also provided with a security type, a communication has an impact on the log too. The tariff output via the standard output (in step 4) leads to a flow of data marked with the type insurance into the standard output channel *stdout*. Since the user data previously flowed into the calculation of the insurance tariff, this is also logged. Information flows into communication channels and persistent memories are hence loggable.

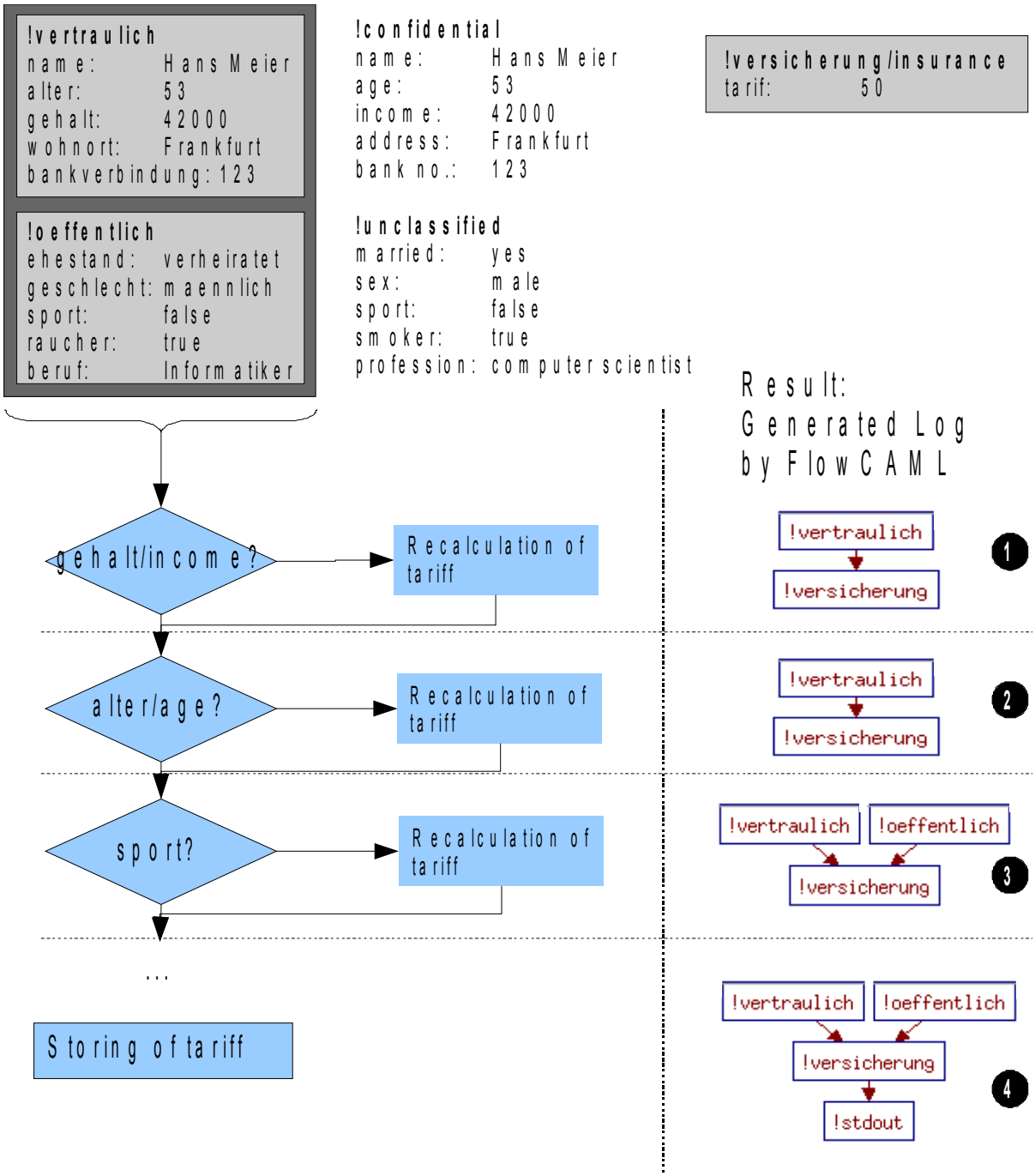

Figure 26 Information flow report expands during the course of the application execution.

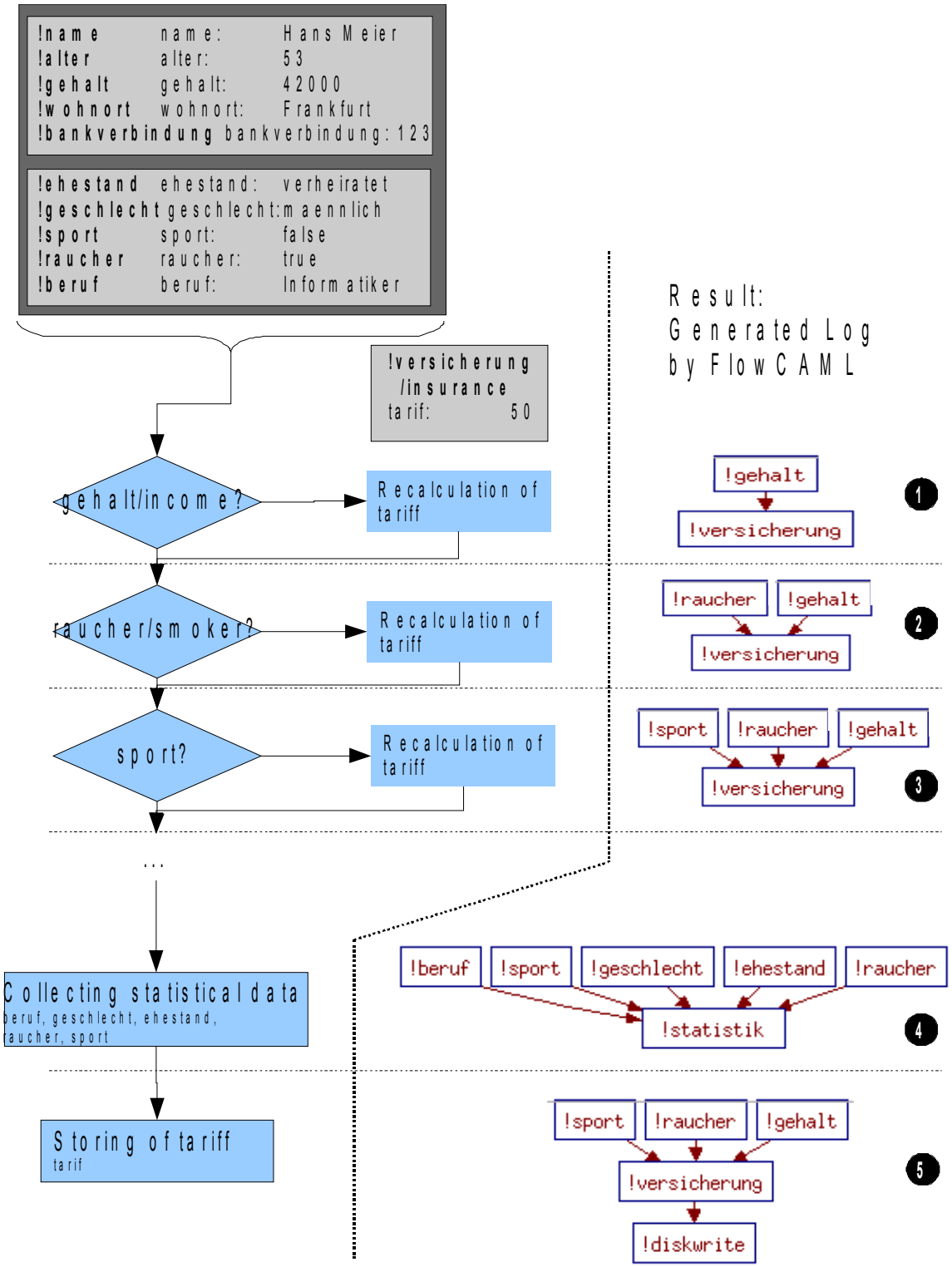

Figure 27 Refined Logging of the Information Flows.

Personal data is provided in turn with security types for storing objects. However, instead of a subdivision of the attributes into the categories of *confidential* and *unclassified*, a more refined subdivision with individual security types for each attribute is introduced. The log of the information flow that has taken place thus reflects the influence of objects through individual attributes and enables a more precise logging. The log of the execution now shows attribute-related information flows accessed that are extended by the attribute type (steps 1 to 3). During the communication of several attributes for the purpose of a statistic collection, the log shows the information flow that does not communicate any data regarded as *confidential* (step 4). Through the storage of the tariff in step 5, data flows from objects with the security type *insurance* to a persistent data carrier that is labelled by the type *diskwrite*. The object concerned was influenced by information from objects with the types *sport, smoker, income*. The log from step 3 is therefore extended by the flow to *diskwrite* (step 5). The concluding log thus consists of entries from steps 4 and 5.

The individual allocation of one security type per attribute shows in detail the information flows through the log. The communicated content can be checked for confidential attributes with data exchange that has taken place and is no longer to be compulsory equated with a potential transmission of data.

At the same time, the information flow analysis shows which data is used and processed for the service provision. No precise assertion about the actual application of an attribute can be derived and it remains unclear whether an attribute in a typed object was transmitted into an object without loss of information[24], or whether the flow materialized through an operation with a reduction of information. However, the context in which typed attributes are used is shown.

6.2.3.5 Conclusion

The application of the information flow analysis presents an effective extension of the encapsulation of non-certified service applications. Suspicious factors created by a communication or storage can thus be avoided. The processing of an application can be made transparent with regard to the requirements. The following prerequisites are necessary for this:

- **Allocation of security types for input and output:** Incoming and outgoing data must be provided with a security type. Particularly for outgoing information flows, e.g. to sub-services, it must be ensured through the encapsulation that a security type is inseparably transmitted with the data. Furthermore, the transmitting communication partner must be convinced that:
    1. with the recipient a service application is present in an execution environment with information flow analysis
    2. security types are continued in his system.

  An execution environment that has these attributes can be authenticated via the TC-authenticated service access points presented.

[24] E.g. a copy or loss-free transformation into another presentation.

- **Standard terminology and semantics of security types:** For the allocation of security types for variables and objects that store personal-related attributes, it must be ensured that the service providers involved and the service user have a common understanding about the significance and use the same terminology of the security types. If no generally accepted ontology is found for this, misunderstandings can arise during the evaluation of the execution protocol.

The type inference detects information flows which take place within a service application and lead out from the application. From the point of view of privacy, not only can threats arise for a person whose confidential data is published without authorization, but also when individual objects that are each classified as non-confidential are merged together as one. Through a combination of this non-confidential data, a specific property vector can be produced which adequately describes a user and is suitable, for example, for recognizing the person concerned (Sweeney, 2002).

It is therefore interesting to examine whether a security type is to be allocated to a quantity of objects typed as non-confidential that is rated higher than the types of the individual objects for identifying a threat. The underlying models of current approaches for analyzing information flows (*FlowCAML, JIF*) do not consider this aspect.

6.2.4 Closing Evaluation of the Encapsulation

The encapsulation of a non-certified service application documents the processing of released data for the user. This enables the user to follow the processing and to examine it with regard to the observance of a user specification. The monitored execution of an application can thereby detect events caused by a storage or transmission of data. The application of protection-worthy data can be fundamentally followed, provided that the information flows in the application can be monitored.

This approach therefore addresses the following aspects:

- Prevention of an non-consented storage or transmission of personal-related data, or creation in this regard of a base data.
- Prevention of an non-consented application of personal-related data, or creation in this regard of a base data.

For this, the execution of a service application in the form of a sealed environment mapped by software is monitored and logged. The presence of the encapsulation can be verified analogous to the presence of a certified service application. The following requirement is therefore also fulfilled:

- Processing of the data in the presence of a mechanism that enables the base for transparency through an audit.

The criterion can also be conditionally fulfilled. It is to be examined here, however, as to whether the restrictions enforceable through an encapsulation guarantee a confidential processing of protection-worthy data in the respective application case.

The encapsulation involves a technical approach which observes activities of the service application through a monitor and generates a log, as well as makes the resultant log

accessible to the user concerned. Through the employment of the TC-authenticated service access points, the user ascertains before service utilization that the service application concerned is executed in an encapsulation which records events and makes them accessible. The approach thereby creates the base for determining by a subsequent evaluation whether the processing of protection-worthy data has taken place in line with a processing guideline of the user. The logging of a service application through a certified encapsulation can thus provide a base data to set up a trust relationship between service user and service provider.

6.2.4.1 Encapsulation of the Service Application

With the encapsulation of an unknown service application without information flow analysis, it appears that communication and storage events are observable. Irrespective of the direction of the data flow, it can be distinguished here more precisely, whether data is flowing to or away. Since no assertion can be made about the contents for the communication or storage, it must be assumed that protection-worthy data is communicated. Therefore, only with the absence of storage and communication events a reliable evaluation of the log can take place. This does not allow an insight into the application to gain information about the application of protection-worthy data. The certified encapsulation of an unknown service application enables the logging of:

- an incoming and outgoing communication and consequently of a potential transmission and
- storage of data.

The observed execution of unknown service applications can, however, technically verify these facts for all applications that do not communicate any data to external applications.

6.2.4.2 Encapsulation with Information Flow Analysis

With an encapsulation that has an information flow analysis, the observation of an unknown application is not just limited to the communication and memory events. The flow of information can be monitored dependent on type through the personal-related data provided with security types.

This means that during a communication and storage, personal data flowing away can be detected on the basis of its security type. The number of suspicious cases that arise through communication and storage operations with an encapsulation without flow analysis can hence be reduced. This particularly benefits applications that communicate with further services for service provision or have to carry out storage operations.

Furthermore, the information flow analysis gives information about the application of protection-worthy data by indicating which data is responsible for influencing a date.

For an unknown application, the certified encapsulation with information flow analysis enables the logging of

- a communication where objects are transmitted,
- the storage of objects and
- insights into the usage of trustworthy data by means of its types.

However, with both encapsulations it is not possible for a user to apprehend when the processing of his data has been completed. If the log of the entire processing is not available, i.e. the log is not complete; this can lead to false conclusions in an evaluation.

6.3 Architecture for privacy-preserving Information Filtering

This section describes an architecture for privacy-preserving information filtering based on the employment of a TCG-compliant platform on the server side. The solution addresses the use case described in section 3.2, but in addition to preserving the privacy of the user data, it takes into account the privacy of the other actors, namely the information service provider and the recommender system provider, resulting in an architecture which preserves privacy in a multilateral way. The architecture is based on Multi-Agent System (MAS) technology because fundamental features of agents such as autonomy, adaptability and the ability to communicate are essential requirements of the chosen approach.

6.3.1 Definitions and Requirements

There are three main abstract entities participating in an information filtering process within a distributed system: A user entity, a provider entity and a filter entity. Whereas in some applications the provider and filter entities explicitly trust each other, because they are deployed by the same party, the described solution is applicable more generically because it does not require this kind of explicit trust between the main abstract entities.

The user is the entity which intends to obtain personalized recommendations, based on private data collected in a user profile. The information these recommendations are based on is collected in the provider profile linked to the information provider entity. The filter entity provides filtering techniques, i.e. the algorithms used to generate the recommendations. The following sections focus on aspects related to the information filtering process itself, and omit all aspects related to information collection and processing, i.e. the stages in which profiles are generated and maintained, mainly because these stages are less critical with regard to privacy, as they involve fewer different entities.

The architecture aims at meeting the following requirements with regard to privacy:

- **User Privacy:** No linkable information about user profiles should be acquired permanently by any other entity or external party, including other user entities. Single user profile items, however, may be acquired permanently if they are unlinkable, i.e. if they cannot be attributed to a specific user or linked to other user profile items. Temporary acquisition of private information is permitted as well. Sets of recommendations may be acquired permanently by the provider, but they should not be linkable to a specific user. These concessions simplify the resulting protocol and allow the provider to obtain recommendations and single unlinkable user profile items, and thus to determine frequently requested information and optimize the offered information accordingly.
- **Provider Privacy:** No information about provider profiles, with the exception of the recommendations, should be acquired permanently by other entities or external parties. Again, temporary acquisition of private information is permitted. Additionally, the propagation of provider information is entirely under the control of the provider.

Thus, the provider should be enabled to prevent misuse such as the automatic large-scale extraction of information.

- **Filter Privacy:** Details of the algorithms applied by the filtering techniques should not be acquired permanently by any other entity or external party. General information about the algorithm may be provided by the filter entity in order to help other entities to reach a decision on whether to apply the respective filtering technique.

Additionally, general requirements regarding the quality of the recommendations as well as security aspects, performance and broadness of the resulting system have also to be addressed. While minor trade-offs may be acceptable, the resulting system should reach a level similar to existing (non-privacy-preserving) recommender systems with regard to these requirements.

6.3.2 Outline of the Solution

The basic idea for realizing a protocol fulfilling these privacy-related requirements in recommender systems is implied by allowing the temporary acquisition of private information: User and provider entity both propagate the respective profile data to the filter entity. The filter entity provides the recommendations, and subsequently deletes all private information, thus fulfilling the requirement regarding permanent acquisition of private information.

The entities whose private information is propagated have to be certain that the respective information is actually acquired temporarily only. Trust in this regard may be established in two main ways:

- **Trusted Software:** The respective entity itself is trusted to remove the respective information as specified.
- **Trusted Environment:** The respective entity operates in an environment that is trusted to control the communication and life cycle 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 not to act in a malicious manner (e.g. it is trusted not to acquire and propagate the respective information itself).

In both cases, trust may be established in various ways. Reputation-based mechanisms, additional trusted third parties certifying entities or environments or Trusted Computing mechanisms may be used. This approach is based on a trusted environment realized via Trusted Computing mechanisms, because this solution constitutes the most generic and realistic approach. This decision and its realization are discussed in section 6.3.4.

The abstract information filtering protocol as shown in Figure 28 consists of the following steps: 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. Through mechanisms provided by this environment, the relay entity is able to control the communication of the TFE, and the provider 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 (steps 1.1 to 1.3 of Figure 28), 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 provider entity from the user entity to the TFE via the relay entity. In the second stage (steps 2.1 to 2.3 of Figure 28), the provider entity establishes control of both relay and TFE, and thus prevents them from propagating provider profile information. Provider profile data is propagated from the provider entity to the TFE via the relay entity. In the third stage (steps 3.1 to 3.5 of Figure 28), 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 problems of determining acceptable queries on the provider profile and ensuring unlinkability of the recommendations are discussed in the following section.

This approach requires each entity in the distributed architecture 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 respective 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 external manipulation attempts, and the ability to control and restrict the communication of other entities.

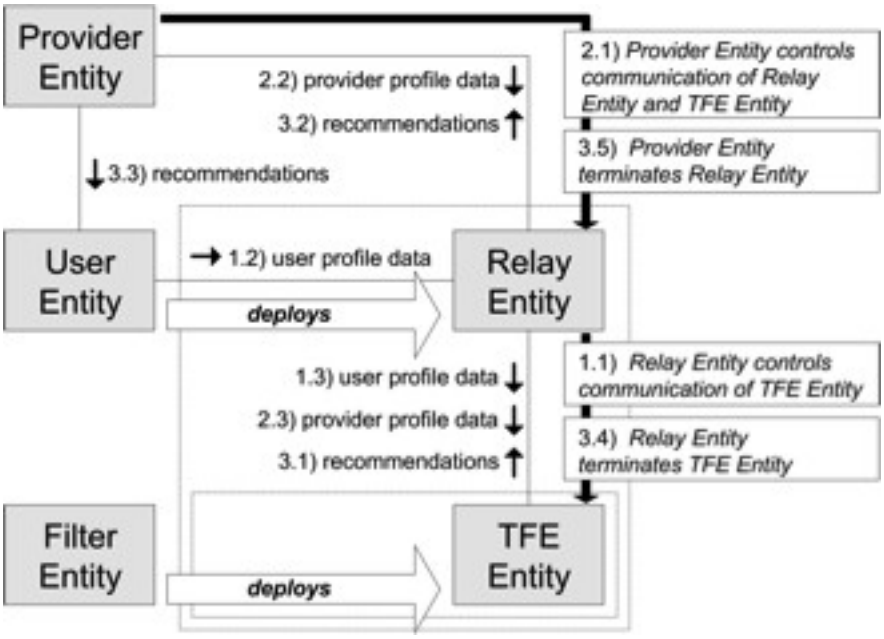

Figure 28 The abstract privacy-preserving information filtering protocol. All communication across the environments indicated by dashed lines is prevented with the exception of communication with the controlling entity.

MAS architectures are an ideal solution for realizing a distributed system characterized by these features, because they provide agents constituting entities that are 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. hosts attacking agents, has to be addressed explicitly. Furthermore, existing MAS architectures generally do not allow agents to control the communication of other agents. It is possible, however, to expand a MAS architecture and to provide designated agents with this ability. For these reasons, this architecture is based on a FIPA-compliant MAS architecture. The entities introduced above are mapped directly to agents, and the trusted environment in which they exist is realized in the form of agent platforms.

In addition to the MAS architecture itself, which is assumed as given, the architecture consists of the following five main modules:

- The *Controller Module* described in Section 6.3.3.1 provides functionality for controlling the communication capabilities of agents.
- The *Transparent Persistence Module* facilitates the use of different data storage mechanisms, and provides 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. Its description is outside the scope of this document.
- The *Recommender Module*, details of which are described in Section 6.3.3.2, provides recommender system functionality.
- The *Matchmaker Module* provides matchmaker system functionality. It additionally utilizes social aspects of MAS technology. Its description is outside the scope of this document.
- Finally, a separate module described in Section 6.3.3.3 provides exemplary filtering techniques mainly in order to show that various restrictions imposed on filtering techniques by this approach may actually be fulfilled. These filtering techniques may be used as the foundation for more advanced algorithms to be applied in real-world recommender systems.

The trusted environment introduced above encompasses the MAS architecture itself and the Controller Module, which have to be trusted to act in a non-malicious manner in order to rule out the possibility of malicious hosts.

6.3.3 Main Modules and Implementation

This section describes the main modules of the architecture for privacy-preserving information filtering, and outlines the implementation. While a specific MAS architecture has been chosen for the implementation, the specification of the module is applicable to any FIPA-compliant MAS architecture. A module basically encompasses ontologies, functionality provided by agents via agent services, and internal functionality. Regarding the terminology used in the following, $\{m\}_{K\_X}$ denotes a message m encrypted via a non-specified symmetric encryption scheme with a secret key K_X used for encryption and decryption which is initially known only to participant X. A key K_XY is a key shared by participants X and Y. A cryptographic hash function is used at various points of the protocol, i.e. a function returning a hash value $h(x)$ for given data x that is both preimage-resistant and collision-resistant[25]. A

[25] In the implementation, Advanced Encryption Standard (AES) has been used as the symmetric encryption scheme and SHA-1 as the cryptographic hash function.

set of hash values for a data set $X = \{x_1, .., x_n\}$ is denoted as $H(X) = \{h(x_1), .., h(x_n)\}$, whereas $h(X)$ denotes a single hash value of the entire data set X.

6.3.3.1 Controller Module

As noted above, the ability to control the communication of agents is generally not a feature of existing MAS architectures but at the same time a central requirement of the described approach for privacy-preserving information filtering. The required functionality cannot be realized based on regular agent services or components, because an agent on a platform is usually not allowed to interfere with the actions of other agents in any way. Therefore, additional infrastructure is added providing the required functionality to the MAS architecture itself, resulting in an agent environment with extended functionality and responsibilities.

Controlling the communication capabilities of an agent is realized by restricting via rules, in a manner similar to a firewall, but with the consent of the respective agent, its incoming and outgoing communication to specific platforms or agents on external platforms as well as other possible communication channels, such as the file system. Consent is required because otherwise the overall security would be compromised, as attackers could arbitrarily block various communication channels. The described approach does not require controlling the communication between agents on the same platform, and therefore this aspect is not addressed. Consequently, all rules addressing communication capabilities have to be enforced across entire platforms, because otherwise a controlled agent could just use a non-controlled agent on the same platform as a relay for communicating with agents residing on external platforms. Various agent services provide functionality for adding and revoking control of platforms, including functionality required in complex scenarios where controlled agents in turn control further platforms. The implementation of the actual control mechanism depends on the actual MAS architecture. In the implementation, methods provided via the Java Security Manager as part of the Java security model have been utilized. Thus, the supervisor agent is enabled to define custom security policies, thereby granting or denying other agents access to resources required for communication with other agents as well as communication in general, such as files or sockets for TCP/IP-based communication.

6.3.3.2 Recommender Module

The Recommender Module is mainly responsible for carrying out information filtering processes, according to the protocol described in Table 1. The participating entities are realized as agents, and the interactions as agent services. Mechanisms for secure agent communication are assumed to be available within the respective MAS architecture. Two issues have to be addressed in this module: The relevant parts of the provider profile have to be retrieved without compromising the user's privacy, and the recommendations have to be propagated in a privacy-preserving way.

The solution is based on a threat model in which no main abstract entity may safely assume any other abstract entity to act in an honest manner: Each entity has to assume that other entities may attempt to obtain private information, either while following the specified protocol or even by deviating from the protocol. Following (Goldreich, 1987), the former case is classified as honest-but-curious behavior (as an example, the TFE may propagate recommendations as specified, but may additionally attempt to propagate private

information), and the latter case as malicious behavior (as an example, the filter may attempt to propagate private information instead of the recommendations).

Retrieving the Provider Profile

As outlined above, the relay agent relays data between the TFE agent and the provider agent. These agents are not allowed to communicate directly, because the TFE agent cannot be assumed to act in an honest manner. Unlike the user profile, which is usually rather small, the provider profile is often too voluminous to be propagated as a whole efficiently. A typical scenario consists of a user profile containing ratings of about 100 movies, and a provider profile containing some 10,000 movies. Retrieving only the relevant part of the provider profile, however, is problematic because it has to be done without leaking sensitive information about the user profile. Therefore, the relay agent has to analyze all queries on the provider profile, and reject potentially critical queries, such as queries containing a set of user profile items. Because the propagation of single unlinkable user profile items is assumed to be uncritical, the information filtering protocol is extended as follows: The relevant parts of the provider profile are retrieved based on single anonymous interactions between the relay and the provider. If the MAS architecture used for the implementation does not provide an infrastructure for anonymous agent communication, this feature has to be provided explicitly:

Phase/ Step	Sender ® Receiver	Message or Action
1.1	*R* ® *F*	establish control
1.2	*U* ® *R*	*UP*
1.3	*R* ® *F*	*UP*
2.1	*P* ® *R*, *F*	establish control
2.2	*P* ® *R*	*PP*
2.3	*R* ® *F*	*PP*
3.1	*F* ® *R*	*REC*
3.2	*R* ® *P*	*REC*
3.3	*P* ® *U*	*REC*
3.4	*R* ® *F*	terminate F
3.5	*P* ® *R*	terminate R

Table 1 The basic information filtering protocol with participants *U* = user agent, *P* = provider agent, *F* = TFE agent, *R* = relay agent, based on the abstract protocol shown in Figure 28. *UP* denotes the user profile with *UP* = {up_1, .., up_n}, *PP* denotes the provider profile, and *REC* denotes the set of recommendations with *REC* = {rec_1, .., rec_m}.

The most straightforward way is to use additional relay agents deployed via the main relay agent and used once for a single anonymous interaction. Obviously, unlinkability is only

achieved if multiple instances of the protocol are executed simultaneously between the provider and different users. Because agents on controlled platforms are unable to communicate anonymously with the respective controlling agent, control has to be established after the anonymous interactions have been completed. To prevent the uncontrolled relay agents from propagating provider profile data, the respective data is encrypted and the key is provided only after control has been established. Therefore, the second phase of the protocol described in Table 1 is replaced as described in Table 2. Additionally, the relay agent may allow other interactions as long as no user profile items are used within the queries. In this case, the relay agent has to ensure that the provider does not obtain any information exceeding the information deducible via the recommendations themselves. The cluster-based filtering technique described in Section 6.3.3.3 is an example for a filtering technique operating in this manner.

Phase/ Step	**Sender ® Receiver**	**Message or Action**
repeat 2.1 to 2.3 for all *up in UP*:		
2.1	F ® R	$q(up)$ (a query based on *up*)
2.2	R^{anon} ® F	$q(up)$ (R remains anonymous)
2.3	P ® R^{anon}	$\{PP_{q(up)}\}_{K\_P}$
2.4	P ® R, F	establish control
2.5	P ® R	K_P
2.6	R ® F	$\{PP\}_{q(UP)}$

Table 2 The updated second stage of the information filtering protocol with definitions as above. PP_q is the part of the provider profile *PP* returned as the result of the query *q*.

Recommendation Propagation

The propagation of the recommendations is even more problematic mainly because more participants are involved: Recommendations have to be propagated from the TFE agent via the relay and provider agent to the user agent. No participant should be able to alter the recommendations or use them for the propagation of private information. Therefore, every participant in this chain has to obtain and verify the recommendations in unencrypted form prior to the next agent in the chain, i.e. the relay agent has to verify the recommendations before the provider obtains them, and so on. Therefore, the final phase of the protocol described in Table 1 is replaced as described in Table 3. It basically consists of two parts (Step 3.1 to 3.4, and Step 3.5 to Step 3.8), each of which provide a solution for a problem related to the prisoners' problem described in (Simmons, 1984), in which two participants (the prisoners) intend to exchange a message via a third, untrusted participant (the warden) who may read the message but must not be able to alter it in an undetectable manner. There are various solutions for protocols addressing the prisoners' problem. The more obvious of these, however, such as protocols based on the use of digital signatures, introduce additional threats e.g. via the possibility of additional subliminal channels. In order to minimize the risk

of possible threats, the described approach uses a protocol that merely requires a symmetric encryption scheme.

The first part of the final phase is carried out as follows: In order to prevent the relay from altering recommendations, they are propagated by the filter together with an encrypted hash in Step 3.1. Thus, the relay is able to verify the recommendations before they are propagated further. The relay, however, may suspect the data propagated as the encrypted hash to contain private information instead of the actual hash value. Therefore, the encrypted hash is encrypted again and propagated together with a hash on the respective key in Step 3.2. In Step 3.3, the key *K_PF* is revealed to the relay, allowing the relay to validate the encrypted hash. In Step 3.4, the key *K_R* is revealed to the provider, allowing the provider to decrypt the data received in Step 3.2 and thus to obtain $H(REC)$. Propagating the hash of the key KR prevents the relay from altering the recommendations to *REC′* after Step 3.3, which would be undetectable otherwise because the relay could choose a key *K_R′* so that $\{\{H(REC)\}_{K\_PF}\}_{K\_R} = \{\{H(REC')\}_{K\_PF}\}_{K\_R'}$. The encryption scheme used for encrypting the hash has to be secure against known-plaintext attacks, because otherwise the relay may be able to obtain *K_PF* after Step 3.1 and subsequently alter the recommendations in an undetectable way. Additionally, the encryption scheme must not be commutative for similar reasons.

The remaining protocol steps are interactions between relay, provider and user agent. The interactions of Step 3.5 to Step 3.8 ensure, via mechanisms similar to those used in Step 3.1 to 3.4, that the provider is able to analyze the recommendations before the user obtains them, but at the same time prevent the provider from altering the recommendations. Additionally, the recommendations are not processed at once, but rather one at a time, to prevent the provider from withholding all recommendations. Upon completion of the protocol, both user and provider have obtained a set of recommendations. If the user wants these recommendations to be unlinkable to him, the user agent has to carry out the entire protocol anonymously. Again, the most straightforward way to achieve this is to use additional relay agents deployed via the user agent which are used once for a single information filtering process.

Phase/ Step	**Sender ® Receiver**	**Message or <u>Action</u>**
3.1	*F* ® *R*	$REC, \{H(REC)\}_{K\_PF}$
3.2	*R* ® *P*	$h(K\_R), \{\{H(REC)\}_{K\_PF}\}_{K\_R}$
3.3	*P* ® *R*	*K_PF*
3.4	*R* ® *P*	*K_R*
repeat 3.5 for all *rec* in *REC*:		
3.5	*R* ® *P*	$\{rec\}_{K\_UR:rec}$

repeat 3.6 for all *rec* in *REC*:		
3.6	*P ® U*	$h(K\_P{:}rec), \{\{rec\}_{K\_UR:rec}\}_{K\_P:rec}$
repeat 3.7, 3.8 for all *rec* in *REC*:		
3.7	*U ® P*	*K_UR:rec*
3.8	*P ® U*	*K_P:rec*
3.9	*R ® F*	terminate F
3.10	*P ® R*	terminate R

Table 3 The updated final stage of the information filtering protocol with definitions as above.

6.3.3.3 Exemplary Filtering Techniques

The filtering technique applied by the TFE agent cannot be chosen freely: All collaboration-based approaches, such as collaborative filtering techniques based on the profiles of a set of users, are not applicable because the provider profile does not contain user profile data (unless this data has been collected externally). Instead, these approaches are realized via the Matchmaker Module, which is outside the scope of this document. Learning-based approaches are not applicable because the TFE agent cannot propagate any acquired data to the filter, which effectively means that the filter is incapable of learning. Filtering techniques that are actually applicable are feature-based approaches, such as content-based filtering (in which profile items are compared via their attributes) and knowledge-based filtering (in which domain-specific knowledge is applied in order to match user and provider profile items). An overview of different classes and hybrid combinations of filtering techniques is given in (Burke, 2002). Two generic content-based filtering approaches have been implemented that are applicable within the described approach:

A direct content-based filtering technique based on the class of item-based top-*N* recommendation algorithms (Deshpande, 2004) is used in cases where the user profile contains items that are also contained in the provider profile. In a preprocessing stage, i.e. prior to the actual information filtering processes, a model is generated containing the k most similar items for each provider profile item. While computationally rather complex, this approach is feasible because it has to be done only once, and it is carried out in a privacy-preserving way via interactions between the provider agent and a TFE agent. The resulting model is stored by the provider agent and can be seen as an additional part of the provider profile. In the actual information filtering process, the *k* most similar items are retrieved for each single user profile item via queries on the model (as described in Section 6.3.3.2, this is possible in a privacy-preserving way via anonymous communication). Recommendations are generated by selecting the *n* most frequent items from the result sets that are not already contained within the user profile.

As an alternative approach applicable when the user profile contains information in addition to provider profile items, a cluster-based approach is provided in which provider profile items are clustered in a preprocessing stage via an agglomerative hierarchical clustering approach.

Each cluster is represented by a centroid item, and the cluster elements are either sub-clusters or, on the lowest level, the items themselves. In the information filtering stage, the relevant items are retrieved by descending through the cluster hierarchy in the following manner: The cluster items of the highest level are retrieved independent of the user profile. By comparing these items with the user profile data, the most relevant sub-clusters are determined and retrieved in a subsequent iteration. This process is repeated until the lowest level is reached, which contains the items themselves as recommendations. Throughout the process, user profile items are never propagated to the provider as such. The information deducible about the user profile does not exceed the information deducible via the recommendations themselves (because essentially only a chain of cluster centroids leading to the recommendations is retrieved), and therefore it is not regarded as privacy-critical.

6.3.3.4 Implementation

The approach for privacy-preserving information filtering has been implemented based on JIAC IV (Fricke, 2001), 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. JIAC IV stands out among MAS architectures as the only security-certified architecture, since it has been certified by the German Federal Office for Information Security according to the EAL3 of the Common Criteria for Information Technology Security standard. 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, and thus provides all security-related functionality required for the described approach. The JIAC IV architecture has been extended by adding the mechanisms for communication control described in Section 6.3.3.1.

6.3.4 Application of Trusted Computing

As described in Section 6.3.2, the architecture is based on a trusted environment realized via Trusted Computing mainly in order to prevent hosts of agent platforms from acting in a malicious manner. The certified service application to be attested is the runtime environment within which agent platforms operate. An entity intending to deploy an agent on a platform operating within this trusted environment should follow the standard remote attestation process, i.e. request the attestation of the platform, verify the authenticity of the attestation result, and evaluate the platform configuration. After successfully carrying out these steps, agents containing private data may be deployed on the platform without the risk of exposing the private data to any other agent running on the platform, or to the platform provider itself.

7 Results

Trusted computing can also be used for a reverse digital rights management. Instead of protecting digital contents of service providers, the reverse deployment of Trusted Computing in combination with a monitor protects personal data according to the agreed processing rules between a user and a service provider. Service providers show their trustworthiness by using a certain monitor which observes the activities of storing and delegating personal data. Trusted computing is thereby used for the attestation of using such a monitor. The deployment of Trusted Computing as specified by the Trusted Computing Group may therefore be used with minor modifications. One necessary modification of the TCG specification is to close the time gap between the attestation of a service application (monitor) and the collection of personal data, since an information system can be modified between these two activities.

This deliverable has shown that the protection of personal data by enforcing agreed privacy policies is done by using TC-attestated service access points and a monitor in combination with mechanisms of information flow analysis. By using TC-attestated service access points, a user can be sure that his personal data will indeed be sent to the attestated service application. The combination with a monitor detects an undesired storing and delegation of personal data.

There are more straightforward ways to implement privacy-preserving information architectures, e.g. by utilizing a centralized architecture in which the privacy-preserving provider-side functionality is realized as trusted software based on Trusted Computing. However, these approaches seem to be unsuitable because they are far less generic: Whenever some part of the respective software is patched, upgraded or replaced, the entire system has to be analyzed again in order to determine its trustworthiness, a process that is problematic in itself due to its complexity. In the described solution of section 6.3 only a comparatively small part of the overall system is based on Trusted Computing. Because agent platforms can be utilized for a large variety of tasks, and because Trusted Computing seems to be the most promising approach to realize secure and trusted agent environments, it seems reasonable to assume that the respective mechanisms will be generally available in the future, independent of specific solutions such as the one described here.

8 Bibliography

Rafael Accorsi. Towards a Secure Logging Mechanism for Dynamic Systems. In: Proceedings of the 7th IT Security Symposium. 2005.

Ammar Alkassar and Rani Husseiki (Eds.). FIDIS deliverable D3.9 "Study on the Impact of Trusted Computing on Identity and Identity Management". FIDIS NoE Consortium – EC Contract No. 507512. 6th Framework Application of European Commission. 2007.

Ross Anderson. Cryptography and competition policy: Issues with "Trusted Computing". In Proceedings of the 22nd Annual ACM Symposium on Principles of Distributed Computing (PODC-03), pages 3-10, New York, July 2003. ACM Press.

Ross Anderson. 'Trusted Computing' Frequently Asked Questions TC / TCG / lagrande / NGSCB / longhorn / palladium / TCPA. World-Wide Web document, August 2003.

ARM. TrustZone Technology - Secure extension to the ARM architecture. http://www.arm.com, 2004.

Paul Ashley, Satoshi Hada, Günter Karjoth, Calvin Powers and Matthias Schunter. Enterprise Privacy Authorization Language (EPAL). http://www.zurich.ibm.com/security/enterpriseprivacy/epal/specification, IBM Research, 2003. Last accessed on May, 21st 2007.

Matthias Bauer, Martin Meints and Marit Hansen (eds.). FIDIS deliverable D3.1 "Structured Overview on Prototypes and Concepts of Identity Management Systems". FIDIS NoE Consortium – EC Contract No. 507512. 6th Framework Application of European Commission. 2005.

Andreas Bogk, Rüdiger Weis and Stefan Lucks. TCG 1.2 - fair play with the 'fritz' chip? In SANE 2004: Proceedings of the 4th International System Administration and Network Engineering Conference, 2004.

Robin Burke. Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction, 12(4):331–370, 2002.

Iris Bohnet and Steffen Huck. Repetition and Reputation: Implications for Trust and Trustworthiness in the Short and in the Long Run. Technical report, KSG, November 2003. Working Paper No. RWP03-048.

Ernie Brickell, Jan Camenisch and Liqun Chen. Direct anonymous attestation. In CCS '04: Proceedings of the 11th ACM conference on Computer and communications security, pages 132-145, New York, NY, USA, 2004. ACM Press.

David Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. Communications of the ACM 24(2). p. 84–88. 1981.

Lorrie Cranor and Paul Resnick. Protocols for Automated Negotiations with Buyer Anonymity and Seller Reputations. Netnomics, 2(1):1-23, 2000.

Lorrie Cranor, Marc Langheinrich, Massimo Marchiori, Martin Presler-Marshall and Joseph Reagle. The Platform for Privacy Preferences 1.0 (P3P1.0) Specification. http://www.w3.org/TR/P3P, April 2002. Last accessed on December 12th 2007.

Mukund Deshpande and George Karypis. Item-based top-N recommendation algorithms. ACM Trans. Inf. Syst., 22(1):143–177, 2004.

Tim Dierks and Cristopher Allen. RFC 2246: The TLS Protocol Version 1.0, 1999.

Enterprise Strategy Group. Liquid Machines: Enterprise Rights Management: A Superior Approach to Confidential DataSecurity, 2006.

Simone Fischer-Hübner and Hand Hedbom (eds.). FIDIS deliverable D12.3 "A Holistic Privacy Framework for RFID Applications". FIDIS NoE Consortium – EC Contract No. 507512. 6th Framework Application of European Commission. 2007.

Stefan Fricke, Karsten Bsufka, Jan Keiser, Torge Schmidt, Ralf Sesseler and Sahin Albayrak. Agent-based telematic services and telecom applications. Communications of the ACM, 44(4), April 2001.

Mark Gasson and Kevin Warwick (eds.). FIDIS deliverable D12.2 "Study on Emerging AmI Technologies". FIDIS NoE Consortium – EC Contract No. 507512. 6th Framework Application of European Commission. 2007.

Oded Goldreich. Zero-knowledge twenty years after its invention. http://citeseer.ist.psu.edu/goldreich02zeroknowledge.html, 2002.

Oded Goldreich, Silvio Micali and Avi Wigderson. How to play any mental game. In Proc. of STOC '87, pages 218–229, New York, NY, USA, 1987. ACM Press.

Adolf Hohl and Alf Zugenmaier. Safeguarding Personal Data with DRM in Pervasive Computing. In Robinson, P., Vogt, H., Wagealla, W., eds.: Privacy, Security and Trust within the Context of Pervasive Computing. Volume 780 of The International Series in Engineering and Computer Science., Springer. 2005.

Alexander Iliev and Sean W. Smith: Protecting Client Privacy with Trusted Computing at the Server. IEEE Security and Privacy, Volume 3, Issue 2 (March 2005), IEEE Educational Activities Department, New York, USA, 20–28.

Uwe Jendricke. Sichere Kommunikation zum Schutz der Privatsphäre durch Identitätsmanagement. Rhombos-Verlag, January 2003.

Rebecca T. Mercuri. On auditing audit trails. Communications of ACM, 46(1):17-20, 2003.

D. Molnar, A. Soppera, and D. Wagner: Privacy for RFID through Trusted Computing (short paper). In S. De Capitani di Vimercati and R. Dingledine, editors, Workshop on Privacy in the Electronic Society (WPES), 2005.

Günter Müller and Sven Wohlgemuth (eds.). FIDIS deliverable D14.2 "Study on Privacy in Business Processes by Identity Management". FIDIS NoE Consortium – EC Contract No. 507512. 6th Framework Application of European Commission. 2007.

Andrew C. Myers and Barbara Liskov. Protecting Privacy using the Decentralized Label Model. ACM Transactions on Software Engineering and Methodology 9. p. 410–442. 2000.

Jan Poritz, Matthias Schunter, Els Van Herreweghen and Michael Waidner. Property based attestation - scalable and privacy-friendly security assessment of peer computers. Technical Report RZ3548, IBM Corporation, 2004.

Alexander Pretschner, Manuel Hilty and David Basin: Distributed usage control. In Communications of the ACM 49(9). Special Issue "Privacy and security in highly dynamic systems". p. 39—44. ACM Press. 2006.

Kai Rannenberg, Andreas Pfitzmann and Günter Müller. IT Security and Multilateral Security. In Günter Müller and Kai Rannenberg (eds.). Technology, Infrastructure, Economy, volume 3 of Multilateral Security in Communications, pages 21—29. Addison Wesley Longman Verlag GmbH, 1999.

Aviel D. Rubin and Daniel E. Geer Jr. Mobile Code Security. IEEE Internet Computing, 2(6):30-34, 1998.

Stefan Sackmann and Jens Strüker. Electronic Commerce Enquête 2005 – 10 Jahre Electronic Commerce: Eine stille Revolution in deutschen Unternehmen. Technical Report, Institute of Computer Science and Social Studies (Telematics), Freiburg I.Br., 2005.

Stefan Sackmann, Jens Strüker and RafaelAccorsi. Personalization in Privacy-Aware Highly Dynamic Systems. Communications of the ACM 49(9), p. 32–38, 2006.

Bruce Schneier and John Kelsey. Security audit logs to support computer forensics. ACM Transactions on Information and System Security, 2(2):159–176, May 1999.

Gustavus J. Simmons. The prisoners' problem and the subliminal channel. In David Chaum (ed.) Proceedings of Crypto '83, pages 51–67. Plenum Press, 1984.

Vincent Simonet. Fine-grained Information Flow Analysis for a lambda-calculus with Sum Types. In: Proceedings of the 15th IEEE Computer Security Foundations Workshop (CSFW 15), Cape Breton, Nova Scotia (Canada). P. 223–237. 2002.

Latanya Sweeney. k-Anonymity: A Model for Protecting Privacy. Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 10. p. 557–570. 2002.

Herbert H. Thompson. Why Security Testing is Hard. Security & Privacy Magazine, IEEE, 1(4):83{86, August 2003.

Transmeta Corporation. Transmeta Announces First Embedded Security Features for x86 Microprocessors. http://investor.transmeta.com/news/20030114-99407.cfm, 2003.

Trusted Computing Group. TCG Backgrounder. https://www.trustedcomputinggroup.org, May 2003.

Trusted Computing Group. Trusted Computing Group TPM Specifcation Version 1.2. https:// www.trustedcomputinggroup.org, 2003.

Trusted Computing Group. Bylaws of Trusted Computing Group. https://www.trustedcomputinggroup.org/about/articles_of_incorporation.pdf, 2003.

Tse, S., Zdancewic, S. Run-time Principals in Information-flow Type Systems. In: IEEE Symposium on Security and Privacy. 2004.

Sven Wohlgemuth and Günter Müller. Privacy with Delegation of Rights by Identity Management. In Günter Müller (ed.), Emerging Trends in Information and Communication Security, International Conference, ETRICS 2006, Freiburg, Germany, June 6-9, 2006, Volume 3995 of Lectures Notes in Computer Science, Heidelberg, Springer, p. 175–190, 2006.

Tatu Ylönen. SSH - Secure Login Connections over the Internet. Proceedings of the 6th Security Symposium USENIX Association: Berkeley, CA:37, 1996.

Alf Zugenmaier and Adolf Hohl. Taking the Ubiquitous Administrator out of the Trust Chain. http://citeseer.ist.psu.edu/676940.html, 2004.