Trust Building and Usage Control for Electronic Business Processes

Vom Fachbereich Informatik der Technischen Universität Darmstadt
genehmigte Dissertation zur Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)
von

Dipl.-Inform. (FH) Dipl.-Betriebsw. (FH)
Omid Tafreschi aus Teheran

Gutachter: Prof. Dr. Claudia Eckert
Prof. Dr. Rüdiger Grimm
Prof. Dr. Stefan Katzenbeisser

Tag der Einreichung: 31.03.2009
Tag der Disputation: 18.05.2009

Darmstadt 2009
Hochschulkennziffer D17

Wissenschaftlicher Werdegang[1]

Seit 3/2009	Wissenschaftlicher Mitarbeiter im Fachbereich Informatik der Technischen Universität Darmstadt
8/2008-2/2009	Wissenschaftlicher Mitarbeiter am Fraunhofer-Institut für Sichere Informations-Technologie
12/2003-7/2008	Wissenschaftlicher Mitarbeiter im Fachbereich Informatik der Technischen Universität Darmstadt
4/1999 - 11/2003	Wissenschaftlicher Mitarbeiter am Fraunhofer-Institut für Sichere Informations-Technologie
9/1999 - 7/2003	Studium der Internationalen Betriebs- wirtschaftslehre an der Hochschule Darmstadt
9/1995 - 4/1999	Studium der Informatik an der Hochschule Darmstadt

Erklärung[2]

Hiermit erkläre ich, dass ich die vorliegende Arbeit - mit Ausnahme der in ihr ausdrücklich genannten Hilfen - selbstständig verfasst habe.

1 gemäß §20 Abs. 3 der Promotionsordnung der TU Darmstadt
2 gemäß §9 Abs. 1 der Promotionsordnung der TU Darmstadt

Aus Sicht von Unternehmen ist die Informationstechnologie (IT) die Schlüsseltechnologie zur Erhaltung ihrer Wettbewerbsfähigkeit. Zu diesem Zweck setzen Unternehmen IT ein, um ihre Geschäftsprozesse zu optimieren. Dabei werden nicht nur interne Geschäftsprozesse, welche innerhalb eines Unternehmens ablaufen, betrachtet, sondern auch zwischenbetriebliche Geschäftsprozesse, an denen mehrere autonome Unternehmen mit Eigeninteressen beteiligt sind, berücksichtigt. Jedoch bringt der IT-Einsatz nicht nur Vorteile mit sich, sondern auch eine Reihe an Schwachstellen und Risiken. Die Digitalisierung von Daten, zum Beispiel Dokumenten, erleichtert deren Bearbeitung und Austausch. Zugleich vereinfacht die Digitalisierung unautorisierte Zugriffe auf sensible Daten. Hoch verfügbare Kommunikationsnetze bieten effiziente Plattformen für Unternehmenskooperationen. Mit diesem Potenzial gehen auch Missbrauchsmöglichkeiten einher, die auf Grund des offenen und anonymen Charakters von Kommunikationsnetzen entstehen. Zum diesen Missbrauchsmöglichkeiten zählt unter anderem Nichteinhaltung von Vereinbarungen, zum Beispiel in Bezug auf die Qualität der gelieferten Güter oder auf Zahlungsmodalitäten.

Als Folge der skizzierten Schwachstellen leidet die Effizienz (die Wertschaffung) eines Geschäftsprozesses oder die Unternehmen verhalten sich zurückhaltend in Bezug auf den IT-Einsatz zwecks Optimierung ihrer Geschäftsprozesse. Vor diesem Hintergrund hat diese Arbeit die Entwicklung von Sicherheitsmechanismen für Geschäftsprozesse zum Ziel. Angesichts der oben genannten Schwachstellen werden die Themen Vertrauensbildung in Kommunikationsnetzen und Nutzungskontrolle für digitale Daten, zum Beispiel elektronische Dokumente, adressiert.

Im Bereich Vertrauensbildung werden zwei neue Reputationssysteme vorgestellt. Das erste Reputationssystem fördert die Vertrauensbildung in einer Kooperationsplattform, welche die Durchführung bilateraler und multiattributiver Verhandlungen ermöglicht. Dabei berücksichtigt das Reputationssystem alle Phasen einer Markttransaktion. Es ermöglicht die Findung eines vertrauenswürdigen Geschäftspartners und bietet anschließend eine Entscheidungsunterstützung während einer Verhandlung. Das zweite Reputationssystem eignet sich zur Vertrauensbildung im Rahmen von Geschäftsprozessen in dezentralen P2P-Netzen. Das System besteht aus einer Reihe von Protokollen, welche die Robustheit des Systems gegenüber Angriffen gegen das Reputationssystem sicherstellen. Ein Schwerpunkt hierbei ist die transitive Berechnung von Vertrauenswerten, welche gegen

gängige Koalitionsangriffe gegen Reputationssysteme robust ist. Diese Eigenschaft des Reputationssystems wird mittels Simulation nachgewiesen.

Das Themenfeld Nutzungskontrolle umfasst zwei Aspekte. Einerseits geht es um die Formulierung von Sicherheitsrichtlinien zur Nutzungskontrolle. Andererseits sind Mechanismen zur Durchsetzung der definierten Sicherheitsrichtlinien notwendig. Diese Arbeit liefert Lösungen zur Formulierung und Durchsetzung von Sicherheitsrichtlinien.

Im Themenfeld Formulierung von Sicherheitsrichtlinien wird ein innovativer Ansatz für die Zugriffskontrolle für strukturierte Dokumente vorgestellt. Das Besondere hierbei ist die Aufzeichnung von ausgeführten Operationen in einer Historie, welche für die Definition von Zugriffsregeln verwendet wird. Das Ergebnis ist ein feingranularer und zugleich flexibler Mechanismus zur Formulierung von Regeln zur Zugriffskontrolle für elektronische Dokumente, die im Rahmen von Geschäftsprozessen ausgetauscht und bearbeitet werden.

Im Bereich Durchsetzung von Sicherheitsrichtlinien wird ein System zur Umsetzung der Nutzungskontrolle für digitale Güter vorgestellt. Es erlaubt die Umsetzung von Geschäftsprozessen im Bereich des Vertriebs digitaler Güter. Das System bietet Anbietern digitaler Güter ein hohes Sicherheitsniveau und ermöglicht gleichzeitig Nutzern flexible Nutzungsmöglichkeiten, zum Beispiel den Transfer von digitalen Inhalten zu anderen Geräten oder Nutzern. Ein wesentlicher Baustein des Systems ist ein robustes Protokoll, welches die Überprüfung der Integrität einer entfernten Plattform, die für die Durchsetzung von Sicherheitsrichtlinien verantwortlich ist, ermöglicht.

ABSTRACT

Information technology (IT) supports companies to streamline their business processes. The main contributions of IT are the digitalization of data and efficient communication networks, which allow companies to automatize their business processes and thus increase their efficiency, i.e., their value creation. This effort started with the optimization of internal business processes within a company. Nowadays, it also includes external business processes, in which multiple enterprises and even customers are involved.

However, using IT also causes undesirable side effects for companies. They are exposed to a wide range of vulnerabilities and threats. Digitalizing data, e.g., documents, spurs the access to that data and the exchange of it. However, a disadvantageous result of digitalizing data is the increased risk of unauthorized access to that data. Communication networks provide an excellent foundation for collaboration between companies. At the same time, the open and anonymous character of communication networks is a reason for distrust towards business partners offering their goods and services over such networks. As a result of these undesirable side effects, the outcome of a certain business process supported by IT may be suboptimal or companies may refrain from using IT. Against this background, this thesis focuses on securing electronic business processes with regard to two aspects, i.e., building trust in open networks and controlling the usage of digital objects.

Trust is the prerequisite for all kinds of commercial transactions. Using reputation information is one possible way to build up trust among business partners. In this thesis, we propose two new reputation systems to establish trust for ad-hoc processes in open markets. The first reputation system facilitates trust building in the context of electronic negotiations which are performed with the help of a centralized system. The reputation system enables companies to find trustworthy business partners and provides decision support during a negotiation. The second reputation system supports trust building in decentralized Peer-to-Peer (P2P) networks. A main feature of this system is its robustness against coalition attacks, which is proven with the help of a simulation.

Controlling the usage of digital objects demands two functionalities. First, we need methods for defining usage rules. Second, mechanisms for enforcing the defined usage rules are required. In this thesis, we address both aspects of usage control.

Digital documents play a central role in business processes, since they are a means of integration and are handled among

business partners. Some documents are sensitive and thus have to be protected from being accessed by unauthorized parties. For this purpose, we propose a flexible and expressive access control model for electronic documents. Our model captures the information about the operations performed on documents. This history information can be used to define access control rules.

Customers are involved in the execution of special kinds of business processes, such as selling and consuming digital goods. In these cases, digital goods have to be protected from being used in an unauthorized way, e.g., being shared in public networks. Thus, the trustworthiness of customers' platforms has to be verified before transferring digital goods. For this, we propose a robust integrity reporting protocol which is necessary when a remote platform has to perform security relevant operations, e.g., to enforce a security policy which controls the usage of digital content. This integrity reporting protocol is a building block of a new Digital Rights Management system which is also presented in this thesis. This system provides a high protection level. At the same time, it allows users to transfer their purchased content to other devices or users.

CONTENTS

LIST OF FIGURES

1

INTRODUCTION

1.1 MOTIVATION

Computers and computer networks revolutionized many areas of our personal and professional life. A wide range of data and services can be accessed and used independently from location or time. From companies' points of view, these possibilities provide an excellent basis for maintaining their competitiveness. In order to exploit these opportunities, companies have started to optimize their business processes with the help of the information technology. This effort is known as Business Process Reengineering (BPR)[HC93]. BPR has many facets along the whole value chain of a company. Electronic markets enable companies to find appropriate business partners at low transaction costs. Supply chain management systems help companies to streamline their cross-organisational business processes and thereby to save their resources. In addition, communication networks like the Internet enable companies to set up a new range of business models, e.g., offering digital content, and to interact directly with their customers. As a result, customers are more and more involved in a company's business processes.

However, using information technology for BPR also causes undesirable side effects for companies. They are exposed to a wide range of vulnerabilities and threats, such as viruses, hacker attacks or data theft. According to [Gar06], 40 percent of all organizations will be targeted by financially motivated cybercrime by 2008. As stated by the computer crime and security survey [Ins07], 194 US companies yielded losses of $66,930,950 caused by various types of computer security incidents in 2007. The vulnerabilities and threats do not only pose major risks to the reliable execution of business processes but may also have negative effects on the reputation of a company and thus on its value [ER02, CGLZ03]. In addition, the vulnerabilities and threats lead to an issue regarding trust building. In contrast to traditional "real-world" transactions, electronic transactions in virtual communities lack the "direct interpersonal" aspects like

mimics, gestures or the personal impression, which have a significant influence on the amount of trust shown towards the business partner. According to [Mui02], challenges facing trust in virtual communities ironically arise from what some proponents claim to be the advantages for such communities: being vast, nearly anonymous, simple to join and leave.

Securing business processes is a challenging task since it has many facets. Information technology itself provides the necessary primitives, e.g., cryptography, to secure business processes at a low level. For example, we can transmit sensitive data over encrypted channels. However, this is necessary but not sufficient, since low level security mechanisms do not support companies and their partners to tap the full potential offered by information technology. In other words, security mechanisms have to be integrated into the business logic of a company. For this purpose, more sophisticated security mechanisms, which ideally cover business processes at application level, are required. Only then, companies can fully optimize their business processes.

1.2 CONTRIBUTIONS

This thesis focuses on securing electronic business processes at application level with regard to two aspects, i.e., trust building and usage control. The contributions and corresponding publications are sketched in the following:

Trust Building

Trust is the prerequisite for all kind of commercial transactions. Using reputation information is one possible way to build up trust between business partners. For this purpose, we propose two new reputation systems to establish trust for ad-hoc processes in open markets. The first reputation system facilitates trust building in the context of electronic negotiations which are performed with the help of a centralized system [RT02, RTT03, TFR05, TMF+07, TMF+08]. The second reputation system supports trust building in decentralized Peer-to-Peer (P2P) networks. A preliminary version of that system has been presented in [Taf07].

Another approach to building trust is to make the actions of a business partner transparent to the others. For this, we analyze a concrete business process called "Sale by Call for Tenders" with respect to trust building. Then, we present a set of protocols improving the fairness of the mentioned business process [TSF+01].

Usage Control

Controlling the usage of digital objects demands two functionalities. First, we need methods for defining usage rules. Second, mechanisms for enforcing the defined usage rules are required. In this thesis, we address both aspects of usage control.

Digital objects play a key role in business processes. For example, a digital document is of utmost importance to a business process, since it is a means of integration and is handled among business partners. Some documents are sensitive and thus have to be protected from being accessed by unauthorized parties. For this purpose, we propose a flexible access control model for electronic documents [RTME06, RTE07b, RTE07a, RTME07, RT08].

Digital objects can embody multimedia content. Since this kind of content, e.g., digital music, can be efficiently reproduced and delivered, a company can set up business models based on such content. However, these technical benefits also enable customers to use their content in an unauthorized way. For instance, they can illegally duplicate and distribute their purchased content, which in turn threatens the business models based on selling digital content. In this thesis, we analyze the requirement for systems protecting digital content [STW04] and present a robust protocol for controlling the usage of digital content [STRE06]. This protocol is a building block for a fair Digital Rights Management system, which is also presented in thesis.

1.3 OUTLINE

This thesis is organized as follows. In Section 2, we give some background information about business processes and argue why it is crucial to secure them. In Section 3, we focus on trust building in the context of business processes. We explain the notions of trust and reputation. Afterwards, we present two new reputation systems which support trust building in open environments. Then, we show how a concrete business process, i.e., Sale by Call for Tenders, can be made robust against classical security attacks, such as identity masquerading and repudiation of messages. In addition, we explain how to overcome economic design problems of the mentioned business process to lower the inhibition threshold for the involved parties. Usage control is the focus of Section 4. There, we first propose a model for controlling access to XML documents, then we analyze the requirements for business processes from the domain of content distribution and usage. Afterwards, we propose a robust integrity reporting protocol for secure content distribution. At the end of Section 4,

we present a fair Digital Rights Management system, which considers the requirements of content providers as well as customers. We conclude with a summary and an outlook into future work in Section 5.

2

BACKGROUND

In this section, we provide an introduction into business processes and investigate security challenges which arise when we try to automatize business processes with the help of information technology. In the following section, we present two concrete application scenarios, in which information technology supports BPR. Afterwards, we use these scenarios to argue why securing business processes is crucial.

2.1 BUSINESS PROCESSES

According to [VW00], no generally accepted definition of the term business process exists since business processes have been approached by a number of different disciplines. [LDL03] claims that most business process definitions are limited in depth and the corresponding models are also constrained to a specific viewpoint. According to [VTT08], the problem with business process definitions in literature is twofold: either they are too simplistic and basic, thus too generic to provide any tangible contribution or they are confined to a very specific application area that prevents them from wide acceptance and applicability.

In this thesis, we use one of the most cited definitions of a business process which is given by Davenport: a business process is "a structured, measured set of activities designed to produce a specific output for a particular customer or market" [Dav92]. In other words, a business process transforms some sort of input by some sort of activities to some sort of output. The goal of this transformation is value creation, i.e., the output of a process is more valuable to an involved party than its input. Davenport's definition clearly shows that business processes embody the core activities of a company which create value for customers and thus ensure the existence of the company. As a result, companies started to pay attention to their business processes in order to increase their efficiency. This effort is referred to as Business Process Reengineering (BPR). Hammer and Champy define BPR as "... the fundamental rethinking and radical redesign of business

processes to achieve dramatic improvements in critical contemporary measures of performance, such as cost, quality, service, and speed" [HC93]. In the last years BPR became a fixed part and success factor of companies [GKS04].

[MS96] states that there are three factors influencing the result of a business process, namely process design, human actors and information technology. Process design defines the way the activities of a business process are actually preformed. For instance, it determines the interdependencies between activities, such as orders and preconditions. Human actors are responsible for performing tasks which cannot be automatized due to their complexity. Thus, the skill and motivation of human actors are decisive factors for the performance of a business process. Information technology provides the means to BPR, since task support and automation is a key approach for BPR. In the following section, we investigate the impact of information technology on BPR.

2.2 THE ROLE OF INFORMATION TECHNOLOGY

At the beginning of BPR, the main focus was on internal business processes within the borders of companies. The objective at that time was to streamline tasks during a business process and to try to automatize them with the help of information technology. Enterprise application integration with the help of enterprise resource planning systems was the prevailing activity at that time. Key technologies were efficient database management systems and distributed client server applications. Later on, uses of information technology extended computer use beyond transaction processing into communication and coordination. As a consequence, the focus of BRP shifted to external business processes. These cover integration patterns across multiple enterprises and interaction with customers. The driving force of this shift was new communication possibilities offered by information technology. Available communication networks, especially the Internet, and open standards, such as the eXtensible Markup Language (XML) have been facilitating collaborative business processes.

At the same time, the digitalization of data does not only allow companies to streamline their business processes, but it also fosters a new kind of business process built up on selling digital media, e.g., digital music and electronic books.

In the following sections, we present two concrete application scenarios, in which the information technology supports BPR. First, we describe an electronic market which enables companies to carry out bilateral and multi-attributive negotiations. In Section

3.3, we propose a reputation system supporting trust building for electronic markets.

In Section 2.2.2, we explain a specific business process, i.e., Sale by Call for Tenders (CFT). A naive translation of the CFT to open networks introduces new manipulation possibilities like identity masquerading, repudiation of messages etc. In Section 3.5, we show how the basic CFT can be made robust against these classical security attacks. However, this approach does not eliminate fundamental economic design problems of the CFT itself. To overcome these shortcomings, we show how the CFT can be protected against manipulations that damage the fairness and economic efficiency of a market by turning it into a secure sealed-bid auction protocol.

2.2.1 *A Business Web Forming an Electronic Market*

Markets provide the basis for almost all types of businesses. According to [Bak98], they have three main functions, i.e., matchmaking between buyers and sellers, providing an institutional infrastructure and the facilitation of transactions. A typical market transaction consists of 5 phases [RL03], which are depicted in Figure 2.1.

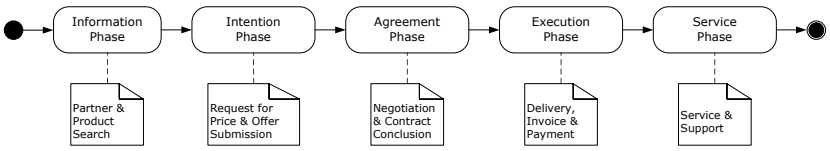

Figure 2.1: Transaction Phases on a Market

In the information phase market participants gather relevant information concerning products, market partners, etc. During the intention phase market participants submit offers concerning supply and demand. Then, the terms and conditions of the transaction are specified and the contract is closed in the agreement phase. The agreed-upon contract is operationally executed in the execution phase. During the service phase support, maintenance and customer services are delivered.

A major activity during the agreement phase is the conduction of negotiations for closing business transactions and concluding valid contracts. A negotiation can be defined as a decentralized decision-making process by at least two parties [Bicoo]. It is performed until an agreement is reached or the process is terminated without reaching an agreement by one or all of the partners involved. The objective of this process is to establish a legally binding contract between the parties concerned, which defines

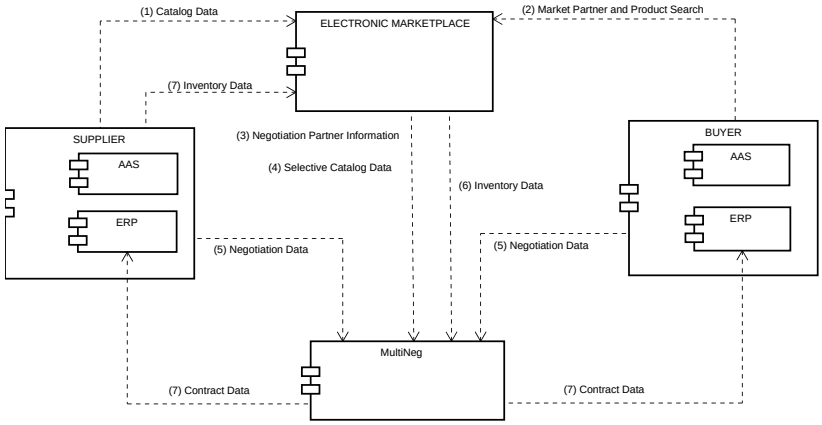

Figure 2.2: Architecture of a SOA-based Business Web

all agreed-upon terms and conditions next to regulations in the case of failure of their fulfillment [RL03]. Beam and Segev define electronic negotiations "as the process by which two or more parties multilaterally bargain resources for mutual intended gain, using the tools and techniques of electronic commerce" [BS97]. Electronic negotiations enable market participants to interact with each other in an ad-hoc manner over large geographical distances.

For the duration of executing a market transaction, all partners involved in its processing, such as buyer, seller, service and infrastructure providers, are inter-linked in a temporary value chain in form of a business web [KR00, TTL00]. Enabling such partner networks to be dynamically created as individually needed can be realized by employing a service oriented architecture (SOA). Thus, the potential partners are not required to have any prior knowledge about the technical requirements for coupling their business applications concerned with processing the market transaction and face no additional preparatory work before communicating electronically.

The MultiNeg Project [Con09] has developed components for forming a SOA-based business web which shapes an electronic market. The market participants can offer their products, negotiate with each other and transfer transaction results into their ERP systems. The business web consists of four kinds of applications, i.e., marketplace, MultiNeg, Authentication and Authorization Server (AAS) and ERP systems. Figure 2.2 shows the overall architecture of the business web and corresponding data flows. The marketplace allows sellers to publish catalogs containing their products or services, which can be subject to negotiations.

MultiNeg is an electronic negotiation support system for bilateral, multi-attributive negotiations. It facilitates the negotiation of

multiple items with arbitrary, variable attributes. The key objectives for the development have been the design of an architecture suitable for different industries, company sizes and products and a communication interface design that allows the integration of inter-organizational with intra-organizational applications. The negotiation functionalities are conceptualized for usage in a decentralized deployment and are based on open standards. Thus, they allow the seamless electronic integration of internal and external business processes.

Several interfaces can be used for the data exchange between the nodes of the business web. They support the exchange of messages with different types of collaborating applications on the inter-organizational as well as the intra-organizational level. Depending on the type of interface, messages hold different operational data like negotiation input, content or status information as well as the negotiation results.

MultiNeg is able to serve as an integration point of intra-organizational planning and allocation processes with inter-organizational market processes. MultiNeg can be applied to any kind of bilateral and multi-attributive negotiations.

The functionalities of MultiNeg are developed for usage in a decentralized deployment and are based on open standards. Companies can use MultiNeg to carry out asynchronous and bilateral negotiations. MultiNeg is a web application which can be accessed by any browser, i.e., it does not require the installation of a special client software. The modular component structure grants functional and administrative application autonomy. Seamless electronic integration of internal and external business processes is achieved through the usage of Web Services. Thus, the system can be used as a stand-alone but may also be used as an add-on for an electronic marketplace system. Figure 2.3 shows the Web Services developed and the messages exchanged between the collaborating applications.

The Init Negotiation Service transfers the data for initializing a new negotiation from the marketplace to MultiNeg. The XML document transmitted is the Negotiation Request. It only requires a confirmation. The Inventory Visibility Service is offered by the supplier as an addition to his catalog published for providing up-to-date inventory information to the buyer in the course of the negotiation. For doing so, the product identification numbers are used for checking the availability of the particular products in demand. This service is used as the back-end service for the Tunnel Inventory Visibility Service, which executes the querying and retrieval of the supplier's stock information requested from the negotiation platform via the electronic market place. There, the

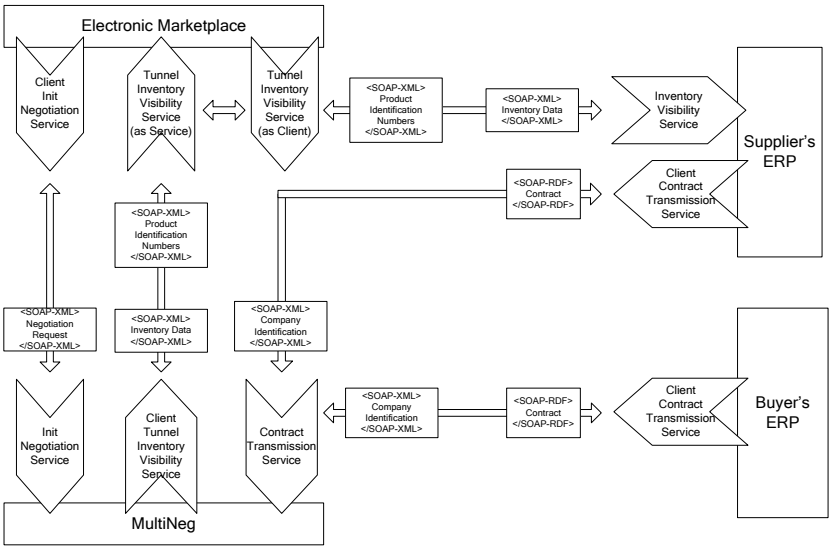

Figure 2.3: Interaction of Web Services

tunnel service aggregates the inventory data returned. Assuming that several suppliers are offering Inventory Visibility Services, this tunnel service is capable of functioning as a composite Web Service of the various single supplier services. After concluding the negotiation, the Contract Transmission Service transfers the agreed transaction details from the negotiation application to the negotiation partners' internal systems. The contract with all details is sent from the negotiation application after having received proper company identification from each partner.

In Section 3.3, we propose a new reputation system for the aforementioned business web. The reputation system enables market participants, i.e., buyers and sellers, to rate each other after a concluded transaction. Using the issued ratings, the market participants can assess the quality of the traded goods as well as the partner's performance. This mechanism enables participants to make better decisions considering the choice of trustworthy business partners and the negotiation strategy.

2.2.2 *Sale by Call for Tenders*

With the emergence of communication networks conventional business processes for trading need to be adapted to the new electronic environment. One such business process is the Sale by Call for Tenders (CFT) which is heavily used for trading perishable goods. The CFT provides a negotiation framework where a seller instantiates a call for tenders, and buyers make buy offers on a competitive basis. The winning buy offer is determined when the CFT is closed. The criterion for determining the winner

can be simply selecting the highest offer, but may also be based on other considerations such as past business relationships with the seller or location. The CFT process can be broken down into four phases. The phases are described below:

1. *Seller calls for tenders* A seller initiates a tender process by submitting lots for sale by way of CFT. More precisely, the seller announces his intention to sell goods, and invites buyers to make purchase offers based on their valuation of the goods. Each lot is described by the following information: lot number, seller reference, branch, sale site organization, product type, product family, product description, unit, quantity, date of availability, place of availability, delivery conditions, and lot price. A seller can modify or delete a sale offer provided no purchase offer has been made on it.

2. *Bid submission* Once a CFT process has started, buyers submit purchase offers on the lots available based on their valuation. This is a simple process where buyers select lots they want to buy and the price which they are willing to pay for each lot. A bid carries the following pieces of information as a minimum: tender Id, buyer reference, quantity, price offered, currency and date and time of bid submission. A buyer can delete or modify an offer provided the closing date has not been reached.

3. *Winner Determination* The selection of winners is done either manually or automatically. For bids to be considered, they must be at least equal to the set reserve price. The reserve price is set by the seller and is known only to him. The criterion for determining the winner is based, among other considerations, on the bid amount where the highest bids stand a greater chance of winning. Once the winners have been selected, the seller contacts them and the tender process is concluded.

4. *Tender Conclusion* This is the last phase of the sale by CFT. Once the winners have been determined, following two statements are distributed: the confirmation statement is sent by the seller to the winners to confirm their orders. The transaction statement describes details of the transaction and the service charges.

The transaction is processed, and payment and delivery executed in line with prior arrangements by the various parties. The actual transfer of goods and money is done outside the system.

A naive translation of the CFT to open networks, e.g. the Internet, introduces new manipulation possibilities like identity masquerading, repudiation of messages etc. In Section 3.5, we show how the basic CFT can be made robust against classical security attacks. However, this approach does not eliminate fundamental economic design problems of the CFT itself. To overcome these problems, we show how the CFT can be protected against manipulations that damage the fairness and economic efficiency of a market by turning it into a secure sealed-bid auction protocol.

2.3 On Securing Electronic Business Processes

A typical business process consists of several tasks which are performed by different entities belonging to different domains. Each party can have its state of knowledge by having more or better information than others as well as its own motives. This characteristic leads to situations with asymmetric information in which individual actions cannot be observed. In economics, an information asymmetry occurs when one party involved in a transaction has more or better information than the other parties. This situation can create an imbalance of power in a business process which again can result in a suboptimal outcome of that business process. Information asymmetry occurs in particular in conjunction with external business processes. According to [WvdHD98], there are two features of external business processes that cross organization boundaries, which distinguish them from internal business processes. First, the resources that are needed for different parts of the process cannot be assigned centrally as they reside in different organizations. Second, the organizations have at least a certain degree of autonomy. These two features together with opportunities offered by information technology lead to undesirable side-effects. Over open and anonymous networks, market participants have to cope with s much higher amount of uncertainty about the quality of products and the trustworthiness of other participants [ZLZ05]. Ackerlof showed that knowledge about the trustworthiness of a seller is vital for the functioning of a market [Ake70]. Therefore, building trust in virtual environments is of high importance. Information about the reputation can be used to build up trust among strangers, e.g., new business partners. Learning from experiences of other users by considering their assessment of a potential partner's business performance as well as the quality of the delivered goods can reduce this uncertainty.

The Sale by Call for Tenders process introduced in the previous section works well in a closed environment with stable business

relationships. Buyers and sellers know and trust each other. In addition, the number of buyers and sellers is small. In virtual environments all these assumptions do not usually hold. Business relationships are much less stable, buyers and sellers can flexibly join and leave a market, and there are virtually no limits to the number of market participants. A naive adoption of the CFT to virtual environments is subject to manipulation. Parties can act under wrong identities, messages can be modified during transmission or a party sending a message within a CFT process may later claim it has not originated from it or may dispute the exact contents of the message.

Documents play a central role in business processes. A document is a means for integration and is exchanged among entities involved in a business process. As a result of BRP, paper documents were more and more replaced by electronic documents. As electronic documents introduce several advantages, such as space saving storage, faster electronic transfer and the possibility to perform a time saving electronic search. However, electronic documents also bring new risks. One such risk is that electronic documents can be stolen more easily. For example, a hacker can steal electronic documents remotely without entering the building where the computer on which the documents are stored is located.

Figure 2.4 shows the amount of loss for several types of computer crime incidents as reported in [GLLR06]. The second largest amount of loss is attributed to unauthorized access to information, which clearly shows the high demand for access control mechanisms, which are mechanisms to restrict access to authorized persons only. In other words, these mechanisms define who is allowed or denied to access which object. The fourth largest amount of loss is caused by the theft of proprietary information. This sum of this type of loss and of unauthorized access exceeds the amount of loss by virus contamination. Since the sketched types of loss can be reduced with access control mechanisms, this highlights the importance of access control even more. Moreover, Figure 2.4 shows that only a relatively small amount of loss is created by outsiders penetrating the system. This indicates that protection mechanisms should focus on inside attackers, which are authorized users of the system.

We emphasize the significance of protecting documents with the help of a concrete application domain. We consider standards, i.e., documents that contain a classification of products or bill of materials. Many standards today are commonly used and open to public use. International organizations like UN/CEFACT intentionally publish their standards and standards' elements freely

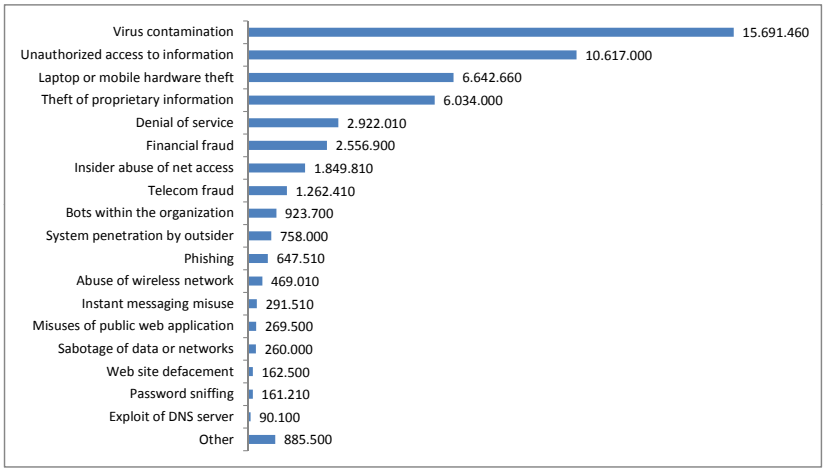

Figure 2.4: Loss in Dollar for Different Types of Incidents [GLLR06]

on the Web to foster their use. Many industry consortiums do the same. Still, many other standards are used without being open to the public or without the public even knowing about them. Thus, access to standards, in part or as a whole, may be restricted. We can distinguish three major reasons for this. First, business models can be based on the commercial use of intellectual property rights for standards, standards' elements or references. Second, some standards are not open to the public, e.g., company-wide internal standards or standards within consortiums. Third, some references are not open to the public, e.g., because those references would unveil ongoing research and development activities.

Against this background, the development of a standard and its elements is an intellectual endeavor that deeply deserves protection of use. If political or strategic considerations do not imply that a standard should be available to the public without restrictions, a company or consortium that has developed a standard may want the option of benefiting commercially from this property.

Instead of using a standard commercially, a company or consortium may want to restrict access to this domain knowledge to users within the company or consortium, e.g., because this knowledge is considered to be a competitive advantage. The same holds true for references between standards. Reasons for not unveiling this information may be similar.

Besides being a competitive advantage or very company specific, a new standard element or reference may also unveil, if made public, information about ongoing research and development or marketing activities to competitors. For example, if a new element is added to a product in a standard, this may alert the competitor that the company plans to add this new feature to the

given product. Or, if in a procurement catalog a new ingredient or part is added, this may reveal that the company is changing recipes or bills of materials and is preparing to market a new or modified product.

For these reasons, access to standards and references between standards may need to be restricted. Therefore, appropriate access-control mechanisms have to be applied.

Business processes can be more or less organized in some aspects. Involved parties performing tasks or the flow of a process, e.g., the order of tasks, may change during the execution of a business process. Singh and Huhns [SH05] state that successful business process management requires supporting three key properties, i.e., autonomy, heterogeneity, and dynamism. Supporting autonomy means modeling and enacting business processes in a manner that offers maximal flexibility to the participants by only minimally constraining their behavior. Supporting heterogeneity means making as few assumptions as possible about the construction of the components for the parties. Dynamism is related to the involved parties which can change during a business process. Supporting these three key properties requires not only hard security, but also soft security. These terms were first described by Rasmussen and Jansson in [RJ96]. They used the term hard security for traditional mechanisms, e.g., access control, and soft security for what they called social control mechanisms, e.g., reputation systems. In Section 3, we present soft security mechanisms supporting trust building in context of business processes. In Section 4, we focus on hard security mechanisms for securing business processes.

3

TRUST BUILDING

In this section, we first motivate the need for reputation systems. For this purpose, we explain the notions of trust and reputation and their interdependencies. We discuss requirements for reputation systems and afterwards present two new reputation systems. The first reputation system supports trust building in a centralized scenario, i.e., electronic negotiations. The second reputation system is designed for decentralized P2P networks. We the analyze a concrete business process, i.e., Sale by Call for Tenders, with regard to trust relationships between involved parties and present two protocols to overcome the existing shortcomings of the mentioned business process. At the end of this section, we review related work and conclude with the summary of our results.

3.1 NOTIONS OF TRUST AND REPUTATION

We rely on trust every day: we trust that drivers on the road will follow traffic rules, we trust that the goods we buy have the expected quality, etc. Generally speaking, trusting becomes the crucial strategy to deal with uncertain, unpredictable and uncontrollable future. This common understanding is founded by numerous definitions of trust. According to [Rip98], trust is "an action that is built upon both, a voluntary provision of a risky service in advance, renouncing any contractual security measure against opportunistic behavior and the expectation that the person whom this service is provided will not act opportunistically." [Szt99] presents a simple and popular definition of trust: "Trust is a bet about the future contingent actions of others."

Most definitions of trust consider persons, such as employees or politicians, as entities to be trusted, i.e., targets of trust. However, there exists other targets of trust. According to [CT01], four different targets of trust are needed and should be modeled and supported:

- trust in the potential partners

- trust in your agent and in mediating agents
- trust in the environment and in the infrastructure (the socio-technical system)
- trust in the authorities

The first three forms of trust targets are of special interest in this thesis. We have to choose the appropriate trust target(s) depending on the characteristics of a business processes. If we consider ad-hoc business processes in which unknown business partners interact with each other, we have two options regarding trust building. First, we directly evaluate the trustworthiness of potential partners. This can be done with the help of reputation systems which are introduced in Sections 3.3 and 3.4. Second, we mandate agents, i.e., Trusted Third Parties, to interact with business partners on behalf of us. We present such an approach in Section 3.5.

[TT00] distinguishes between party trust and control trust: "In a given situation the trading parties can either directly trust each other or rely on functional equivalent control mechanisms, i.e., the procedures and protocols that monitor and control the successful performance of a transaction." If we consider well organized business processes with known but untrusted partners and well established technical environments, we can replace the necessary trust in persons, i.e., business partners, by building trust in the technical environment and in the infrastructure. In this case, we substitute party trust by control trust. We focus on this approach in Section 4. In the following, we focus on party trust.

In contrast to traditional "real-world" transactions, electronic transactions in virtual communities lack the direct interpersonal aspects like mimics, gestures or the personal impression, which have a significant influence on the amount of trust shown towards the business partner. According to [Mui02], challenges facing trust in virtual communities arise ironically from what some proponents claim to be the advantages of such communities: being vast, nearly anonymous, simple to join and leave.

The analysis by Akerloff in 1970 on the Market for Lemons is also applicable to the electronic markets [Ake70]. The main issue pointed out by Akerloff about such markets is the information asymmetry between buyers and sellers. Sellers know about their own trading behavior and the quality of the products they are selling. On the other hand, buyers can at best guess the behavior of one specific seller. Market participants can reduce this information asymmetry by evaluating the reputation of their potential business partners.

The common understanding of the term "reputation" is the perception that one entity has about another's intentions and norms. Reputation has been used in a number of disciplines such as economics and computer science, among others. Economists used reputation to explain "irrational" behavior of players in repeated economic games [Wil85]. Computer scientists have used reputation to model the trustworthiness of entities in online marketplaces [ZMM99]. Figure 3.1 depicts the creation of reputation and its influence on trust. The reputation of one entity is

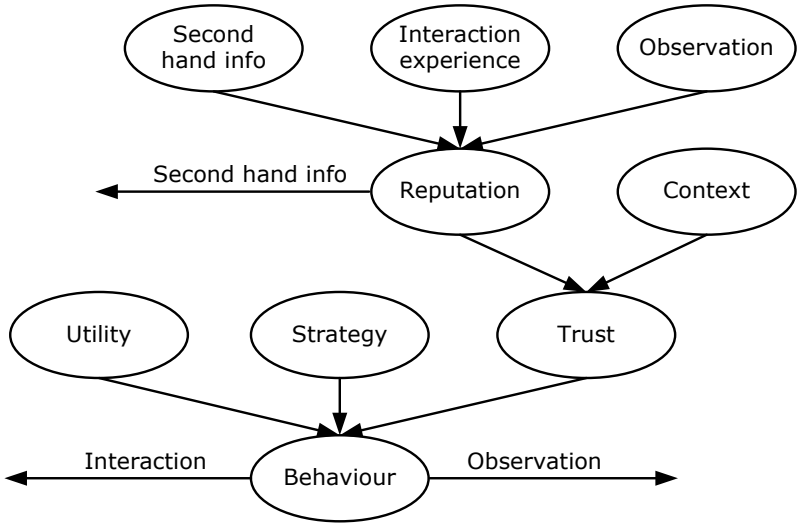

Figure 3.1: Relationship between Reputation and Trust

subjectively formed by others on the basis of their own direct experiences or observations. Second hand information has also an influence on reputation. This kind of information is often passed through "word of mouth", e.g., personal recommendations. Within a certain context reputation may lead to trust or mistrust, which again is the starting point for a decision making process concerning a certain kind of behavior. This decision depends also on the utility and the strategy of the decision maker and may lead to an interaction or further observation.

Reputation is useful in two aspects. First, it helps a market participant with regard to decision making. This is necessary when a buyer has to choose between several sellers offering the same product with comparable conditions. Second, it makes opportunistic behavior unprofitable, since this would lead to a bad reputation, which in turn may deter potential business partners from considering the entity with the bad reputation.

According to [RKZF00], a reputation system collects, distributes, and aggregates feedback about participants' past behaviors. Feedback is usually collected at the end of each transaction. The authority responsible for reputation management asks the participants to submit a rating. Aggregation of feedback means that the available information about an entity's transactions is evaluated and condensed into a few values that allow users to easily make a decision. Feedback is distributed finally to the participants, i.e., it has to be made available to everyone who wants to use it for making decisions on whom to trust. Figure 3.2 shows the components and actors of a reputation system [SVB05].

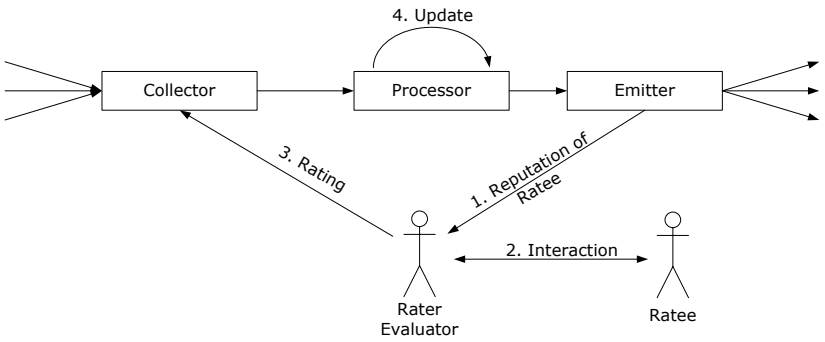

Figure 3.2: Components and Actors of a Reputation System [SVB05]

The target of a rating is called ratee. The collector gathers ratings from agents called raters. This information is processed and aggregated by the processor. The emitter makes the results available to other requesting agents.

One prominent reputation system is eBay's feedback service. eBay buyers and sellers are encouraged to rate one another at the end of each transaction. A rating can be positive, negative or neutral and may include a short text comment. eBay makes the ratings submitted for each member and their aggregation available to all its users. [RZ02] found out that eBay's feedback service has managed to provide remarkable stability in an otherwise very risky trading environment. Dellarocas ascertains that eBay's feedback service is successful to a large extent because people trust eBay [Del01].

[JIB07] distinguishes between centralized and decentralized reputation systems. Centralized reputation systems have dedicated collectors, processors and emitters. In contrast, decentralized reputation systems do not assume any particular entities to play the role of collectors, processors and emitters.

3.2.1 *Requirements for Reputation Systems*

As mentioned above, the main goal of a reputation system is to take ratings which have been given to an entity after concluded transactions, aggregate them and spread the result to produce information about the reputation of the entity in question. [CH07] presents a comprehensive security analysis for reputation systems and identifies the following security requirements for reputation systems:

AVAILABILITY A reputation system, such as the eBay's feedback service, can be a building block of an overall system. In such cases, the availability of the reputation system has to be ensured to prevent the reputation system from becoming a single-point of failure.

INTEGRITY OF REPUTATION INFORMATION The reputation information, i.e., the ratings, should be protected from unauthorised manipulation, both in transmission and in storage.

ACCURACY The reputation system should be accurate with respect to the calculation of ratings.

USABILITY AND TRANSPARENCY Evaluators should be able to comprehend how reputation information is obtained and what it means.

FAIRNESS The values of mutual ratings should be independent from each other to reflect the objective opinion of their raters.

ACCOUNTABILITY Each entity should be accountable for his actions. This property is necessary to prevent raters from issuing unfair ratings.

VERIFIABILITY An evlauator should be able to assign a rating to the corresponding transactions. This property allows evaluators to verify the correctness of the rating in question.

PERFORMANCE EFFICIENCY The reputation system should require as few resources for computation, storage and communication as possible.

In decentralized environments, e.g., P2P networks, there is no central instance which could act as a collector, a processor and an emitter of a reputation system. As a consequence, an adequate reputation system for decentralized environments must fulfill some special requirements. [ZH06] defines the following requirements for decentralized reputation systems:

HIGH ACCURACY To help distinguish reputable peers from malicious ones, the system should calculate the reputation scores as close to their real trustworthiness as possible.

FAST CONVERGENCE SPEED The reputation of a peer varies over time. The reputation aggregation should converge fast enough to reflect the true changes of peer behaviors.

LOW OVERHEAD The system should only consume limited computation and bandwidth resources for peer reputation monitoring and evaluation.

ADAPTIVE TO PEER DYNAMICS Peers join and leave an open P2P system dynamically. The system should adapt to these peer dynamics instead of relying on predetermined peers.

ROBUST TO MALICIOUS PEERS The system should be robust to various attacks by both independent and collective malicious peers.

SCALABILITY The system should be able to scale to serve a large number of peers in term of accuracy, convergence speed, and extra overhead per peer.

The requirement "robust to malicious peers" needs further explanation. According to [MGM06], malicious peers are one of two primary types of adversaries in P2P networks. The other type are the selfish peers. The two types of adversaries are distinguished by their goals. Selfish peers wish to use resources while minimizing their own contribution. Selfish peers are often referred to as "free-riders" [FC05]. Malicious peers aim at causing harm to either other specific peers or the system as a whole. In the context of this thesis we focus on malicious peers.

Attacks performed by malicious peers, though not proper attacks in the sense of information security, threaten the integrity — with respect to their goals — of the informational content, i.e., ratings, stored in the reputation system. One obvious attack in P2P networks is the issuing of fake ratings. These are ratings which do not belong to a real transaction. Especially in decentralized environments without any controlling entities, e.g., P2P networks, the issuing of fake ratings is a feasible attack.

Dellarocas [Del00] presents two kinds of unfair behavior of malicious entities. First, a seller colludes with a group of buyers in order to be given unfairly high ratings to improve his reputation. This collusion is referred as ballot stuffing. Second, sellers and buyers collude to rate other sellers unfairly low to drive them out of the market. This unfair behavior is named bad-mouthing.

Reputation systems which allow mutual ratings, i.e., both involved business partner rate each other after the end of the transaction, should prevent that one party can learn the rating he is supposed to receive before he has issued the rating for his business partner. This should ensure that a rating reflects its issuer's experience in the context of the corresponding transaction and it is independent from the rating of the opposite side. For the same reason, the rater should not be able to modify his rating after it was given to the ratee.

Since the overall reputation of one party depends on all ratings which were given to him, it should not be possible for that party to suppress some of his ratings, i.e., his poor ratings, in order to improve his reputation.

3.3 A REPUTATION SYSTEM FOR ELECTRONIC NEGOTIATIONS

In this section, we present a new centralized reputation system for the business web mentioned in Section 2.2.1. It enables companies to find trustworthy business partners and provides decision support during a negotiation. In order to achieve these two objectives, our reputation system provides the necessary data, i.e., ratings, whenever it is required for an evaluator and offers different aggregations of this data to simplify the assessment of a business partner. One of these aggregations reflects a transitive view from the perspective of the evaluator to the entity to be evaluated. This approach helps evaluators to detect malicious high (or low) ratings which have been issued to improve (or damage) the reputation of a certain entity.

Our reputation system enables market participants, i.e., buyers and sellers, to rate each other after a concluded transaction. They can assess the quality of the traded goods as well as the partner's performance. The issued ratings enable participants to make better decisions considering several aspects. For example a seller is offering products of excellent quality, but tends to deliver late. This information is valuable for potential buyers who have to plan their production in advance. If they decide to buy some goods from the seller in question, they have to schedule reserve time for maybe late delivery. Our reputation system encompasses two phases, i.e., rating phase and rating aggregation phase. Figure 3.3 depicts the rating phase and the rating aggregation phase during a market transaction which have been explained in Section 2.2.1.

In the rating phase two business partners evaluate each other after a concluded transaction by giving a rating to the opposite side. This phase affects the last three transaction phases during which two business partners interact with each other on a mar-

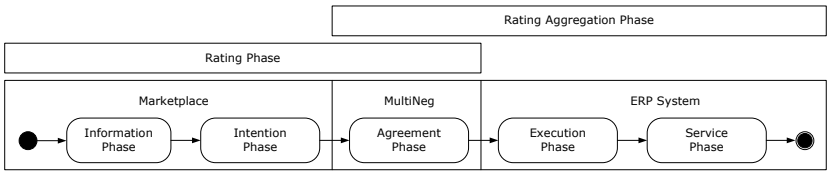

Figure 3.3: Rating Phase and Rating Aggregation Phase During a Market Transaction

ket. During these three phases, including the negotiation, they form an opinion about each other, which is later reflected by the corresponding ratings. These ratings are used in the rating aggregation phase to get information about the reputation of a market participant. The rating aggregation phase takes place during the first three transaction phases. In the information phase and the intention phase a buyer wants to learn more about his potential negotiation partners, i.e., sellers. For this purpose he has to evaluate the rating of sellers. During the actual negotiation both buyer and seller can use the ratings of their opposite to map out their negotiation strategy. We explain the steps which are performed by our reputation system during the rating phase and the rating aggregation phase in the following.

3.3.1 *Rating Phase*

Our reputation system has to fulfill three requirements at the end of the rating phase. First, it has to prevent fake ratings. This requirement means, that ratings should be issued after conducted transactions. Second, the reputation system must ensure the authenticity and integrity of ratings. Third, the system has to make sure that the two ratings issued after a transaction are mutually independent, i.e., they reflect the real experience of the rater. Figure 3.4 shows the process of a negotiation and the subsequent rating phase. We describe the process in the following:

Steps 1-2 After concluding a negotiation the involved parties evaluate each other by issuing corresponding ratings. A rating ranges from 0, i.e., poor rating, to 1, i.e., excellent rating. We support two different types of rating, i.e., buyer rating and seller rating. A buyer rating is given by a seller to a buyer, whereas a seller rating is given by a buyer to a seller.

Steps 3-5 After the receipt of the rating data MultiNeg forwards it to the AAS of the corresponding party. The AAS displays the rating to its issuer and allows him to digitally sign it. The signed rating is transferred to the marketplace to be published.

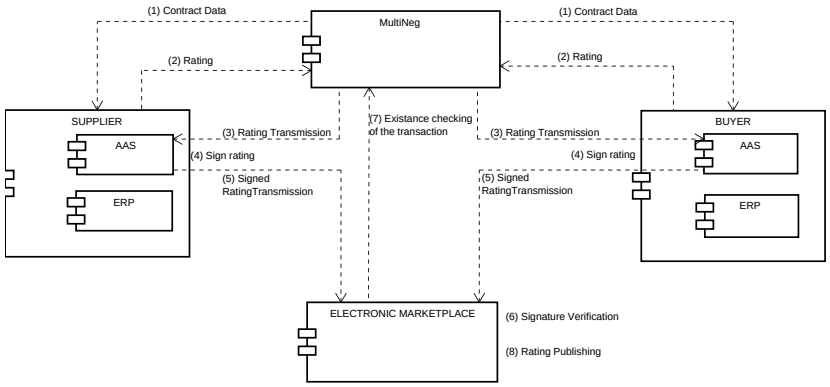

Figure 3.4: Rating Phase

Steps 6-8 The signature of a signed rating is verified at the marketplace. In the case of a positive result, the marketplace checks whether the rating belongs to an existing negotiation. This step ensures that only authentic ratings are published. If MultiNeg's answer is positive, the marketplace publishes the rating if both ratings of a negotiation have been issued. Publishing two ratings simultaneously prevents that one negotiation partner changes his mind (rating) after learning the rating given to him by his opposite. However, this could protect one party from receiving a poor rating. In this case the party does not give a rating for his opposite. To overcome this shortcoming the marketplace publishes a single rating without the second corresponding rating after a defined period of time has passed. This rule is known to all market participants.

3.3.2 *Rating Aggregation Phase*

Existing ratings of market participants are stored and displayed at the marketplace where buyers search the catalogs of sellers and choose partners for a negotiation. The ratings of market participants can be used to support that decision. Our reputation system provides a set of different views to the ratings. In order to simplify the assessment of a certain market participant based on his ratings, our system allows calculating the arithmetic average of all his ratings. Although this approach is easy to implement and to understand, it is not robust against ballot stuffing and bad-mouthing, since it considers only the number of ratings, independent from their issuers. In the following, we show how to overcome this shortcoming.

If we assume that trust is transitive, we can apply the concept of Web of Trust for the calculation of trust values based on rating lists. Pretty Good Privacy (PGP) was the first popular

system which used the concept of Web of Trust to establish the authenticity of the binding between a public key and a user in a decentralized and open environment, e.g., the Internet [Zim95].

We use the concept of Web of Trust to aggregate ratings in our reputation system. For this purpose, the evaluator first has to build up a trust network between himself and the peer for which he wants to calculate the trust value. The trust network is a directed graph and should consist of one trust path at least. A trust path embodies a chain of trust relationships. Each chain link reflects a trust relationship between a rater and a ratee. The start vertex of a trust path is the evaluator and the end vertex is the market participant to be evaluated. In addition, a trust path contains other vertices, i.e., market participants which have been rated by the evaluator or by other vertices in the trust network except the end vertex. In order to keep the calculation base meaningful, we limit the length of a trust path, i.e., the number of its vertices [JHP06]. Each edge of a trust network has a weight, i.e, the rating which was given by the start vertex of the corresponding edge to the vertex at the end of that edge. In the case of several ratings, we use their arithmetic average as weight.

We explain the algorithm for the aggregation of ratings based on Web of Trust with the help of an example. One buyer (Company A) enters the marketplace and wants to purchase products which are offered by the seller (Company Z). These two market participants have never negotiated with each other. But there are other companies which have negotiated with both, seller and buyer. The corresponding ratings are shown in Figure 3.5.

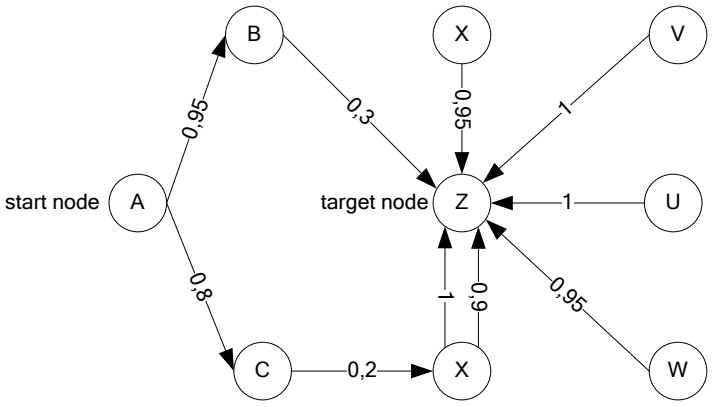

Figure 3.5: Trust Network Containing all Ratings

The figure shows an example for ballot stuffing. Company Z has received 6 unfair high ratings, which have been issued to improve its reputation. It has only one poor rating issued by Company B, which reflects its real experience with Company Z. If Company A only examines the arithmetic average of Company

Z's ratings (0,871), it would consider Company Z trustworthy. In this case, the malicious behavior of Company Z would remain undetected. To overcome this shortcoming, we use the concept of Web of Trust to calculate trust values. For this purpose, we use the existing ratings to build up a trust network between buyer and seller. If there are several ratings between two parties we aggregate them into one value, which is the arithmetic average. In our example, we aggregated the ratings issued by Company X. Figure 3.6 shows the corresponding trust network of our example. This trust network is the input for calculating the trust value for

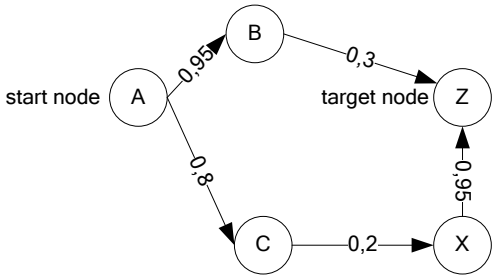

Figure 3.6: Reduced Trust Network

Company Z from the perspective of Company A. For this, we use the following algorithm proposed by Caronni [Caroo]:

Input: trust network W, start node A, target node Z
Output: TrustValue

```
1  TrustValue = 0;
2  trust in A = 1;
3  for all direct neighbors of Z do
4      direct neighbor is X;
5      if X is the start node then
6          TrustValue = 1 − (1 − (trust in X)·
7          (edge_{X,Z}) · (1 − TrustValue));
8      else
9          create a copy C of W;
10         drop target node Z in C;
11         returnValue = calculateTrust(C,A,X);
12         TrustValue =
           1 − (1 − returnValue · edge_{X,Z}) · (1 − TrustValue);
13 return TrustValue;
```

Algorithm 1: Transitive Rating Aggregation [Caroo]

The algorithm shown above has three input parameters, i.e., trust network W, start node A and target node Z. The initial trust value in target node Z is initialized as 0 (line 1) and the trust value in start node A is set to 1 (line 2). Creating a copy of the

trust network (line 10), which is used for the recursion (line 11), assures that the computational results to derive the trust in one neighbor do not influence the trust in any other neighbor of Z. Even though the algorithm starts close to Z (line 3), it will first compute the trust in direct neighbors of the start node, due to the recursion (line 11), and from there move on to derive the trust into their neighbors, and so on until the target node is reached. We apply Algorithm 1 on our example trust network depicted by Figure 3.6. The intermediate steps and their results are shown in the following:

calculateTrust (W,A,Z)	
direct neighbor is B	line 4
calculateTrust (W,A,B)	line 11
direct neighbor is A	line 4
$0,95 = 1 - (1 - 1 \cdot 0,95) \cdot (1 - 0)$	line 6
return $0,95$	line 13
$0,285 = 1 - (1 - 0,95 \cdot 0,3) \cdot (1 - 0)$	line 12
direct neighbor is X	line 4
calculateTrust (W,A,X)	line 11
direct neighbor is C	line 4
calculateTrust (W,A,C)	line 11
direct neighbor is A	line 4
$0,8 = 1 - (1 - 1 \cdot 0,8) \cdot (1 - 0)$	line 6
return $0,8$	line 13
$0,16 = 1 - (1 - 0,8 \cdot 0,2) \cdot (1 - 0)$	line 12
return $0,16$	line 13
$0,394 = 1 - (1 - 0,16 \cdot 0,95) \cdot (1 - 0,285)$	line 12
return $0,394$	line 13

The direct neighbors of Company Z are Company B and Company X. The algorithm starts with Company B, which is not the start node. As a consequence, Company Z has been removed from the trust network and company B is the new target node. The algorithm calls itself with the new trust network. The direct neighbor of Company B is Company A (start node). Line 6 of the algorithm contains the formula for calculating the trust level of Company A with regard to Company B. The calculated value (0,95) is returned from the recursive call. It is the input in the formula in line 12 of the algorithm, where we calculate the trust level of Company A with regard to Company Z considering the rating of Company B. The result is 0,285.

We repeat the sketched process again for Company X as next neighbor of Company Z. Again, we first remove Company Z from the trust network and set Company X as the new target node. The

algorithm proceeds with a recursive call. Since Company X is not a direct neighbor of Company A, we have to remove Company X and set Company C as the target node. The new trust network is the input for a recursive call, which calculates the result level of Company A with regard to Company C (0,8). We proceed to calculate the trust level of Company A with regard to Company Z taking into account the ratings of Company C and Company X. The result is 0,16. The final result reflecting the trust value for Company Z from the perspective of Company A considering all intermediate ratings is 0,394. This value is presented to Company A in addition to the arithmetic average of Company Z's ratings which is 0,871. The two values are contradictory and provide additional information for decision-making with regard to the perspective of the evaluator.

The aggregation of ratings based on the concept of Web of Trust does not totally prevent ballot stuffing and bad-mouthing. However, our example shows that these kinds of malicious behavior lose some of their impact. Recall that ballot stuffing requires that a (small) set of raters give a large number of unfair good ratings to a ratee in order to improve his reputation. Since the approach based on Web of Trust considers the average of ratings given to a market participant, it defuses the impact of several unfair ratings. The same reasoning applies to bad-mouthing. In Section 3.4.8.2 we analyze the robustness of the aggregation of ratings based on the concept of Web of Trust.

The reputation system explained in this section is centralized. That means, it relies on the marketplace as a trusted third party acting as collector, processor and emitter. In the next section, we propose a decentralized reputation system supporting trust building in P2P networks.

3.4 A ROBUST REPUTATION SYSTEM FOR P2P NETWORKS

The reputation system presented in the previous section can be used in scenarios in which trusted parties are available. However, we cannot always take the availability of trusted parties for granted. This case occurs in decentralized P2P networks, in which no dedicated entities exist. As a consequence, we cannot use the proposed centralized reputation system in P2P networks. We need another type of reputation system which is adapted to the special characteristics of P2P networks. In the following section, we first give an overview of P2P networks and their traits. Then, we present a reputation system which takes the traits of P2P networks into account. Afterwards, we prove the robustness of this reputation system with the help of simulation.

3.4.1 *Overview of P2P Networks*

Schoder and Fischbach define Peer-to-Peer (P2P) as a technology "that enables two or more peers to collaborate spontaneously in a network of equals (peers) by using appropriate information and communication systems without the necessity for central coordination" [SF03]. This definition looks at P2P from an application point of view and it emphasizes that the principal focus of P2P is building a network, in which peers can communicate.

Steinmetz and Wehrle define P2P as a "system with completely decentralized self-organization and resource usage" [SW05]. This definition postulates that P2P is a totally distributed system, in which all nodes are completely equivalent in terms of functionality and tasks they perform. This definition fails to encompass, for example, systems that employ the notion of "supernodes", i.e., nodes that function as dynamically assigned localized mini-servers, such as Kazaa, which are, however, widely accepted as P2P [ATS04].

A more general definition of P2P networks is given by Rousopoulos et al. in [RBR+04]. They state that P2P networks have three characteristics: self-organization, symmetric communication and distributed control.

SELF-ORGANIZATION Nodes organize themselves into a network through a discovery process. There is no global directory of peers or resources.

DISTRIBUTED CONTROL Peers determine their level of participation and their course of action autonomously. There is no central controller that dictates behavior to individual nodes. This dynamics of peer participation is referred to as churn [SR06].

SYMMETRIC COMMUNICATION Peers are equals. They request and offer services, rather than being confined to either client or server roles.

These three characteristics lead to the potential benefits of P2P networks. These are among others cost reduction, improved scalability and reliability, and increased autonomy [MKL+02]. However, the same characteristics cause some problems in P2P networks. Due to the lack of any dedicated instance responsible for coordination and control tasks and the dynamic of P2P networks, a wide range of security issues can occur. An overview of security issues in P2P networks can be found in [Wal03]. In [BLP05], Balfe et. al argue that existing security services and mechanisms which have been developed for client-server systems cannot be used in

P2P systems. In traditional client-server systems, users are typically identified with a user account, and system-specific controls can be built on these accounts to provide security services, such as access control and non-repudiation. In addition, client-server systems are normally built up on central entities, which can act as trusted third parties supporting security mechanisms. However, these assumptions do not hold in decentralized P2P networks. As a consequence, another type of security mechanisms is needed for decentralized environments. We present such a mechanism, i.e., a decentralized reputation system, in the following.

3.4.2 *Usage Scenario*

We explain our reputation system with the help of one concrete scenario. This scenario reflects a typical business process between two autonomous peers in commercial settings. Basically, one peer can have two roles in our P2P network. It can be an expert who sells his knowledge or a customer who is willing to pay for the expert's service. We refer to all commercial interactions between customers and experts as transactions. A transaction consists of four phases, which are depicted in Figure 3.7.

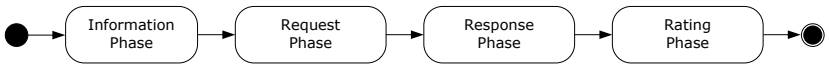

Figure 3.7: Transaction Phases

INFORMATION PHASE In this phase a customer searches for an expert who is able to provide a specific service. If the customer finds some experts with whom he had no direct interactions before, he may want to learn more about their trustworthiness. In this case the reputation of the experts in the P2P network is crucial for the customer. Therefore, the customer wants to learn more about potential business partners. For this purpose, the customer sends queries into the P2P network.

After finding the suitable expert the customer asks the expert for the specific service. At this stage, the expert may wish to learn more about the customer and he can use the same mechanisms like the customer to get some reputation information about the opposite side.

REQUEST PHASE If the expert agrees to deliver the service the customer sends his request to the expert in this phase.

RESPONSE PHASE In this phase the expert provides the service to the customer.

RATING PHASE In this phase the two parties rate each other. The customer may evaluate the quality of services provided by the expert and the expert may assess the payment practice of the customer.

3.4.3 *Basic Idea*

The main challenge for a reputation system for P2P networks is to cope with the characteristics of that kind of networks which have been described in Section 3.4.1. The lack of any central instances responsible for certain control and coordination tasks, i.e., acting as collector, processor and emitter in context of a reputation system, is the main problem. Our approach for solving this problem is based on the introduction of two additional peers, i.e., witnesses, who are involved in a transaction between a customer and an expert. These two peers are ordinary peers. They appear as witnesses and have three tasks. First, they testify the authenticity of a transaction. This should prevent fake ratings. Second, they ensure the fairness of mutual ratings. That means, that both customer and expert have to issue their ratings before they are published. And the third task of the witnesses is to publish the rating lists of the customer and the expert at the end of the corresponding transaction. Generally speaking, the two witnesses act as collector, processor and emitter of a reputation system in the context of one single transaction. Figure 3.8 illustrates the involvement of the witnesses in a transaction.

Both, the customer and the expert choose their own witness. Each witness has to observe the actions of the opposite side, i.e., customer's witness observes the expert and makes a transcript as a proof of interaction. The transcript is a record of all messages which were received by a witness as well as all results which were produced by that witness, e.g. the result of a signature verification. Observing the customer's actions and making a corresponding transcript is the task of the expert's witness. This assignment of the observation tasks to the witnesses makes malicious behavior difficult, since the expert or the customer cannot collude with the witness which was chosen by him and he does not know the witness which is supposed to observe him in advance. We assume that peers which are involved in a transaction stay online until the transaction is completed.

Associating a history of behavior with a particular peer requires a sufficiently persistent identity. We assume that each peer

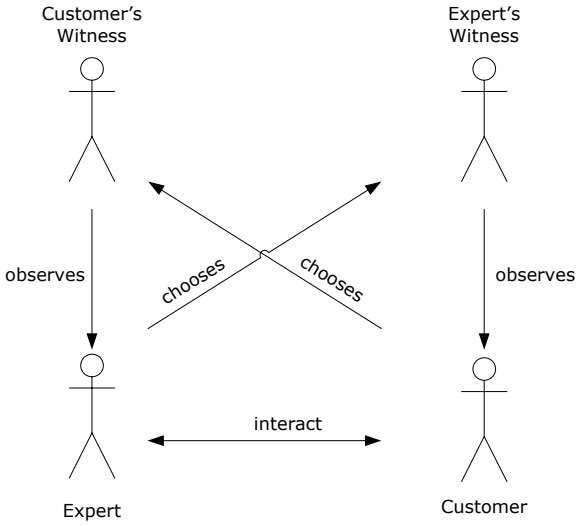

Figure 3.8: Basic Model

has a cryptographic key pair and a digital certificate issued by a trusted Certification Authority (CA). This assumption hampers the decentralized character of a P2P network. However, alternatives, i.e., decentralized solutions, usually require identifiers that are costly to produce, though not strictly unforgeable. Costly identifiers help to slow the rate of whitewashing or generating multiple identities, but do not eliminate it [Dou02].

In the following sections, we first explain the operations and data structures of our reputation system. Then, we present four protocols for the phases depicted by Figure 3.7.

3.4.4 *Operations and Data Structures*

Figure 3.9 lists the cryptographic operations which are used in the protocols of our reputation system.

Encrypting a message m with the key k	$Enc(k, m)$
Decrypting a message m with the key k	$Dec(k, m)$
Signing a message m with the key k	$Sig(k, m)$
Verifying the digital signature of signed message m	$Ver(m)$
Computing the hash value of a message m	$Hash(m)$

Figure 3.9: Basic Cryptographic Operations

As mentioned above, we assume that each peer has a cryptographic key pair, i.e., secret key SK and public key PK. We denote the owner of a certain key by adding his initials as an index to that key, e.g., PK_{CW} is the public key belonging to the witness of the customer.

Two peers (expert and customer) rate each other after a concluded transaction. We define two data structures for this purpose: transaction rating and rating list. The latter consists of all transaction ratings of a specific peer P and has the following structure:

$$\begin{aligned} RatList_i^P &:= \{Rat_1^{E_1,C_1}, \ldots, Rat_i^{E_i,C_i}\} \\ \forall j \in \{1,\ldots,i\} &: E_j \neq C_j; (P = E_j) \vee (P = C_j) \end{aligned}$$

The rating list $RatList_i^P$ contains i transaction ratings of a specific peer P, which he has received after the end of each transaction, in which he was involved either as an expert or as a customer. This condition is denoted by $E_j \neq C_j$. Storing all transaction ratings for P independently from his role in a transaction delivers a broader information about P's behavior, since this view reflects one of the key features of P2P networks, namely, symmetric communication.

The data structure for a transaction rating after the i-th transaction is defined as follows:

$$\begin{aligned} Rat_i^{E,C} := \quad & Sig(s, (\\ & Sig(SK_C, Rat^{C\to E}), \\ & Sig(SK_E, Rat^{E\to C}), \\ & Sig(SK_{EW}, Transcript_{EW}), \\ & Sig(SK_{CW}, Transcript_{CW}), h)) \\ s \in & \{SK_{EW}, SK_{CW}\} \\ h \in & \{Hash(RatList_{i-1}^E), Hash(RatList_{i-1}^C)\} \end{aligned}$$

The whole transaction rating is signed using the key s of the witness which publishes the rating. This entity is either the customer's witness CW or the witness who was chosen by the expert EW. This condition is denoted by $s \in \{SK_{EW}, SK_{CW}\}$. A transaction rating contains following elements:

- the signed rating which was given by the customer to the expert
  $Sig(SK_C, Rat^{C\to E})$
- the signed rating which was given by the expert to the customer
  $Sig(SK_E, Rat^{E\to C})$

- the transcript which was written and signed by the customer's witness
  $Sig(SK_{CW}, Transcript_{CW})$
- the transcript which was written and signed by the expert's witness
  $Sig(SK_{EW}, Transcript_{EW})$
- the hash value of the former rating list of either the customer or the expert
  $h \in \{Hash(RatList^{E}_{i-1}), Hash(RatList^{C}_{i-1})\}$

The transaction rating contains both the signed rating of the customer as well as the signed rating of the expert for two reasons. First, this approach allows the evaluator of a transaction rating to get more information about the corresponding transaction, since he knows also the "reverse of the medal". Second, removing a transaction rating from the rating list becomes a risky action for a malicious peer, since the corresponding reputation list of the business partner may contain the two signed ratings which have been removed. This would lead to a discrepancy between the two rating lists. As a consequence, the cheating of the malicious peer would be detected.

Similar to the signed ratings, the contained transcripts deliver also valuable information for a potential evaluator. In addition, they prove the existence of the corresponding transaction.

Since different peers may be witnesses of different transactions of one specific peer, i.e., customer or expert, we have to ensure that all ratings of that peer are tied together like a chain. For that we add a hash value to the transaction rating. This hash value serves as a pointer to the last rating list of a peer. This pointer allows to build a chain of all transaction ratings of a certain peer. It helps to check the integrity of rating lists, since removing any transaction rating apart from the last ratings from the rating list would breach its integrity. In Section 3.4.7, we present an approach for detecting a manipulated rating list from which the last ratings have been removed.

Figure 3.10 shows a simplified transaction. Details for all phases are presented in Sections 3.4.8-3.4.7. We assume that a rating can range between 0 and 10, where 0 is the poorest and 10 is the best rating. Before the transaction begins, the rating list of the customer contains three transaction ratings and the expert's rating list has five ratings. The rating lists are exchanged during the information phase. Each peer calculates a reputation value which is based on a rating list and reflects the trustworthiness of the rating list's owner. A peer can use different algorithms for this purpose. We discuss this in Section 3.4.8. We assume

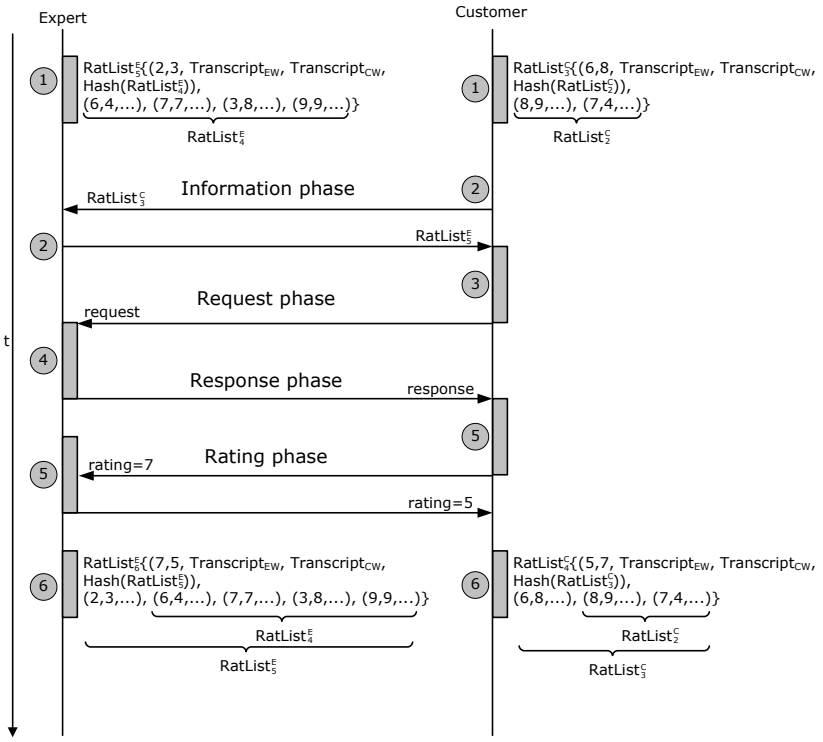

Figure 3.10: Simplified Transaction

that both partners decide to start the transaction. The customer sends his request to the expert who in turn sends a response to the customer. In the rating phase, the business partners rate each other. In our scenario, the customer rates the performance of the expert with 7 and he gets a rating of 5. The new transaction ratings are added to the rating lists. The new transaction ratings contain pointers to the last rating lists. For example, the new transaction rating in the updated rating list of the customer ($RatList^C_4$) has a hash value of the last rating list ($RatList^C_3$).

The sketched approach binds ratings to the course of a specific transaction. This could lead to delays, when a peer has to wait for the end of a transaction, i.e., his rating, before he can start a new transaction. Particularly, very active peers, e.g., powersellers, would be negatively affected by the delays caused by the reputation system. In Section 3.4.7, we show how to solve this issue.

In the following, we describe the protocols for all four phases shown in Figure 3.7 in detail. Since ratings of a peer are aggregated in the information phase, we describe this phase in the end.

3.4.5 *Request Phase*

In this phase, the customer sends his request to the expert. The challenge in this phase is to involve the witnesses into the process of transferring the request from the customer to the expert. At the end of this phase both witnesses should know that the expert has received the customer's request without knowing the request itself.

Our approach for fulfilling these requirements is that the expert has to interact with the witness in order to get access to the customer's request. This has to take place in a way that no one of the parties involved can cheat. The expert should not be able to deny that he has received the same request which was sent by the customer. Vice versa, the customer should not be able to deny that he has sent a certain request to the expert. Figure 3.11 depicts our protocol for the request phase.

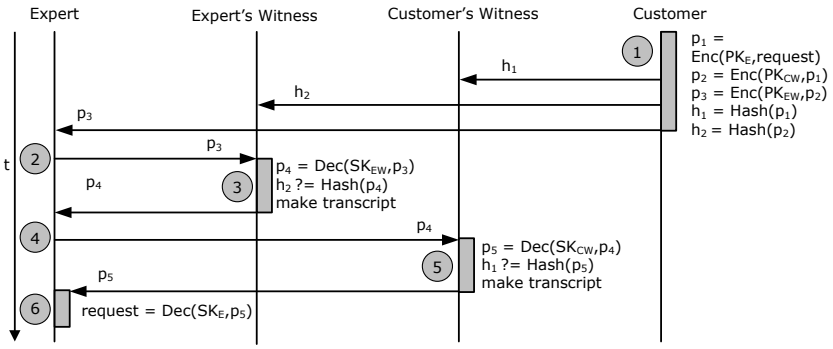

Figure 3.11: Protocol for Submitting the Request

At the beginning of this phase the customer encrypts his request first for the expert $\mathrm{Enc}(\mathrm{PK}_E, request)$. The resulting cipher text p_1 is then encrypted for the customer's witness $\mathrm{Enc}(\mathrm{PK}_{CW}, p_1)$. The resulting cipher text p_2 is again encrypted for the expert's witness $p_3 = \mathrm{Enc}(\mathrm{PK}_{EW}, p_2)$. Note that the order of the encryption steps is significant, since it defines the order of interaction steps performed by the expert. p_3 is sent to the expert. However, he cannot read it; he has to interact with the witnesses first. In other words, the expert asks the two witnesses to decrypt the encrypted request. This approach enables the witnesses to observe the interaction between the customer and the expert.

We have to ensure that the witnesses are able to verify the authenticity of the encrypted packages presented by the expert. For this purpose, the customer computes two hash values of packages which were encrypted for the witnesses h_1 and h_2. These hash values are sent to the corresponding witnesses.

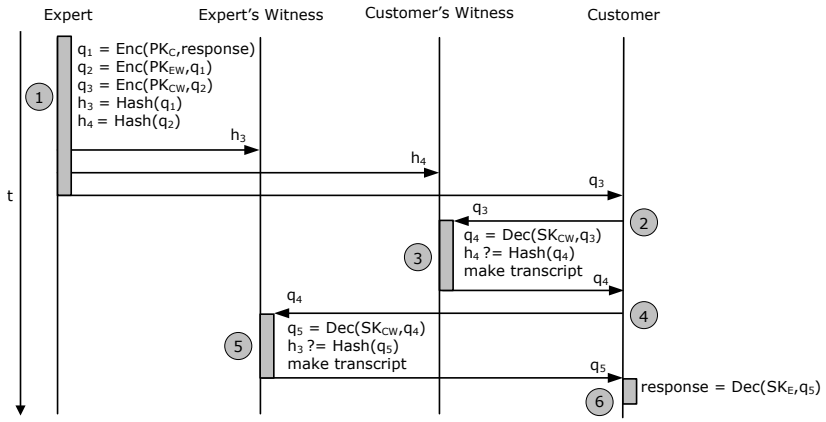

Figure 3.12: Proctol for Submitting the Response

After receiving p_3 the expert has to interact with the witnesses to get access to the customer's request. Since the outer encryption of p_3 can be decrypted only by the expert's witness the expert sends p_3 to his witness. The witness of the expert decrypts p_3 and computes the hash value of the result p_4 with h_1 which he received from the customers. If the two hash values are equal the expert's witness knows that he just decrypted the package which was sent from the customer to the expert. The expert's witness makes a transcript of his actions and sends p_4 to the expert. The latter has to interact with the customer's witness to remove the outer encryption of p_4. He again sends p_4 to the customer's witness who first decrypts the package, compares the hash value of the result p_5 with h_2, makes a transcript and sends p_5 to the expert. Now the expert is able to decrypt the encrypted request with his private key.

3.4.6 *Response Phase*

Figure 3.12 depicts our protocol for the response phase. Since the protocol steps are similar to those in the request phase we refrain from explaining them again.

3.4.7 *Rating Phase*

In this phase the customer and the expert rate each other. Three requirements have to be fulfilled during and after this phase. First, it should be prevented that one party can learn the rating he is supposed to receive before he has issued the rating for his business partner. This should ensure that a rating reflects its issuer's experience in the context of the corresponding transaction

and is independent from the rating of the opposite side. Second, the rater should not be able to modify his rating after it was given to the ratee. This should prevent the rater from acts of revenge. Third, we have to ensure the integrity and authenticity of the ratings, i.e., the new transaction ratings should be added to the corresponding rating lists as described in Section 3.4.4.

We describe the protocol steps for the rating phase which are depicted by Figure 3.13. At the beginning, the two witnesses sign their transcripts. For this purpose, the expert's witness performs the operation $Sig(SK_{EW}, Transcript_{EW})$ and the customer's witness signs his transcript $Transcript_{CW}$ accordingly. After signing the witnesses distribute their signed transcripts to the involved parties. This is necessary since the signed transcripts are going to be contained in the new transaction ratings. The receivers of the signed transcripts have to verify both its signature and its content.

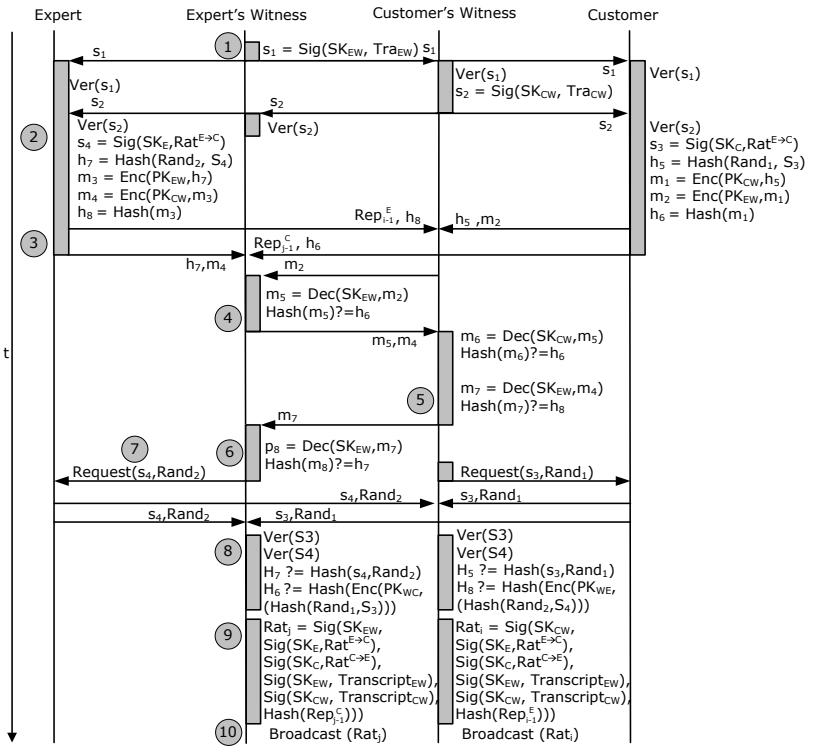

Figure 3.13: Protocol for Rating

If the expert and the customer agree with the transcripts they begin to issue the rating for the opposite side. The basic idea for this purpose is that the rater first seals his rating and commits to the sealed rating. After both ratings have been issued the raters unseal their ratings.

We consider only the actions of the expert, since the customer takes according steps. The expert first signs his rating

$Sig(SK_E, Rat^{E\to C})$ to guarantee its integrity and authenticity. Then, the expert computes the hash value of the signed rating result s_4 together with a random number $Rand_2$. The random number is used to prevent known-plain-attacks since a rating is an element belonging to a finite set. The computed hash value h_7 is encrypted for the expert's witness. The encryption result m_3 is again encrypted for the customer's witness. This multiple encryption forces the witnesses to interact with each other during the progress of the protocol. This interaction is necessary to ensure both the commitment of the raters and the simultaneous publishing of the transaction ratings. After the multiple encryptions the expert computes the hash value h_8. The expert sends h_8 together with his current rating list Rat^E_{i-1} to the customer's witness. Sending the current rating list by its owner is necessary to ensure its consistency in the case of concurrent transactions. With this approach, the owner of a rating list can coordinate the updates of his rating list and avoid inconsistent rating lists. Afterwards, the expert sends m_4 to his witness. In order to remove the outer encryption of m_4 the expert's witness forwards it to the customer's witness. The latter decrypts m_4 with his private key SK_{CW}. The result is m_7. The customer's witness sends m_7 to the expert's witness if the hash value of m_7 is equal to h_8 which was sent by the expert. After receiving m_7 the expert's witness decrypts it with his secret key $Dec(SK_{WE}, m_7)$ and compares the hash value of the result m_8 with h_7.

Similar steps are carried out on the customer's side. At this time both the customer and the expert have committed themselves to their ratings which are only known to them at this time. The witnesses ask the business partners to unseal their signed ratings by sending them and the according random numbers to them.

Again, we explain the next protocol steps from the perspective of the expert. He sends his signed rating for the customer s_4 and the random number $Rand_2$ to the witnesses. Each witness has to carry out tasks which we can divide into three categories, i.e., verifying ratings, updating rating list and publishing updated rating lists.

The expert's witness verifies the signatures of the signed ratings of the customer $Ver(s_3)$ and of the expert $Ver(s_4)$. Then he checks the integrity of the commitments (h_6 and h_7) he has received from the business partners. In the case of the customer's commitment h_6, the expert's witness checks whether h_6 is equal to $Hash(Enc(PK_{WC}, (Hash(Rand_1, s_3))))$. If the result is true, the expert's witness knows that the customer has sent the same rating which was sent earlier by him. After verifying the ratings, the expert's witness creates a new transaction rating for the cus-

tomer. This has the structure which was described in Section 3.4.4.

After creating the new transaction ratings the witnesses broadcast them to the P2P network. This broadcasting is significant, since it exploits one of the main characteristics of P2P networks, i.e., churn introduced in Section 3.4.1, to make malicious behavior difficult in such networks. At the time of broadcasting new transaction ratings, a peer does not know all the other peers which are connected to the network. Any connected peer could receive the broadcast message and store it. As a consequence, the owner of a rating list does not know the other peers who also could have his rating list. Therefore, presenting a manipulated rating list, i.e., a list containing only the first good transaction ratings, is a risky malicious action which can be detected. In order to check the integrity of a rating list, the evaluator can ask connected peers for the ratings of the peer to be evaluated. If the evaluator receives ratings which have not been presented by the owner of the rating list, he can conclude that the presented rating list has been manipulated. In this case, the evaluator would refrain from interacting with the malicious peer.

3.4.8 *Information Phase*

In this phase, a customer looks for experts who are able to provide a certain service. He may wish to get some reputation information of possible candidates by evaluating their reputation lists. These can be stored by the customer itself or by another peer including the experts in question. It it obvious that storing a rating list by only its owner simplifies malicious behavior. For example, an expert who first behaved in an expected way and received good transaction ratings and then started to deliver poor quality and received bad ratings as a consequence could present his old reputation list with good rating to new customers. His true reputation would remain undetected. To overcome this shortcoming, we proposed the broadcasting mechanism which prevents this kind of malicious behavior in Section 3.4.7. The effect of that mechanism is that rating lists are spread out across the P2P network and are stored by several peers.

After getting the rating lists the customer has to draw the reputation information about the owners from the rating lists. We refer to this reputation information as trust value. Calculating one trust value is done by aggregating of transaction ratings. The algorithm to be applied for this purpose is chosen by the evaluator. He can aggregate the ratings according to the approach which has been described in Section 3.3.2 and apply Algorithm

1 proposed by Carroni [Caroo]. We evaluate the robustness of this approach against malicious behavior, i.e. ballot stuffing and bad-mouthing, with the help of a simulation which is explained in the following.

3.4.8.1 *Simulating Malicious Behavior*

In this thesis, we consider malicious peers who collude either to discredit the reputation of an honest peer to benefit from it later on (bad-mouthing) or to advertise the qualities of their own services more than their real values with the help of unjustified high ratings (ballot stuffing). This kinds of behaviors are attacks against the reputation system. As stated in [LZY05], "different attackers usually have different intents even when they issue the same attack." For example, a hacker may want to demonstrate his hacking capacity, whereas business competitors may attack each other's information systems to gain market share. [LR04] describes how malicious market participants try to shill recommender systems in order to promote their own goods and services. This behavior is similar to ballot stuffing attacks against reputation systems. We believe that the robustness against ballot stuffing and bad-mouthing is crucial in context of business processes. In the following, we focus on simulating of these kinds of attacks.

Our simulation allows us to setup a P2P network with two kinds of peers, i.e., honest peers and malicious peers. The size of each peer group is set with the help of a simulation parameter. The simulation runs through several steps. All peers can interact with others at each step. This behavior is also defined with three parameters. We can define (a) the probability for interaction, (b) the value of rating which is given after a transaction depending on the opposite peer's type and (c) the minimum trust value necessary for interacting with a certain peer depending on its type. However, malicious peers always interact with each other independently from trust values. In the following, we describe the results of rating aggregations after ballot stuffing and bad-mouthing attacks.

3.4.8.2 *Ballot Stuffing*

Figure 3.14 depicts a network after 30 steps. The trust network consists of 20 malicious peers (red vertices) and 80 honest peers (green vertices). An edge between two peers indicates an interaction between them and its weight reflects the average value of ratings which have been given by the start vertex to the end vertex. For instance, peers 4 and 70 interacted with each other.

Since peer 4 is a malicious peer and performs poorly, it gets a poor rating (0.1) from peer 70. On the other hand, peer 4 gives a poor rating (0.01) to peer 70. This behavior is part of a predefined rating schema. Honest peers give good ratings to each other and poor ratings to malicious peers. Malicious peers also give good ratings to each other in order to improve their reputation which is the objective of ballot stuffing. They give good or poor ratings to honest peers in order to confuse in the case of a good rating or to pretend that the whole transaction did not fulfill the expectations on either side in the case of a poor rating.

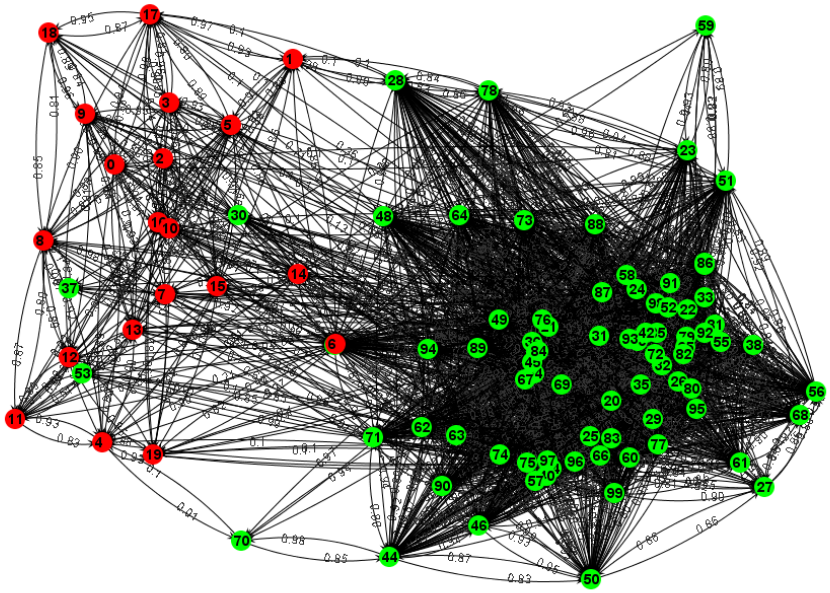

Figure 3.14: P2P Network after 30 Steps

We computed trust values based on the ratings in the trust network shown above with Algorithm 1. The result is depicted by Figure 3.15.

The figure shows from the perspective of all honest peers, i.e., peer 20 to peer 100, the average trust value calculated for a certain peer type at a certain step. The red surface shows the trust values for malicious peers and the green surfaces shows the trust values for honest peers. It is obvious that the red surface converges towards the green surface with the increasing number in transactions. In other words, the average trust value for a malicious peer increases depending on the number of transactions. This trend leads to an indistinguishability between honest and malicious peers. We explain the reason for this phenomenon in Section 3.4.8.4.

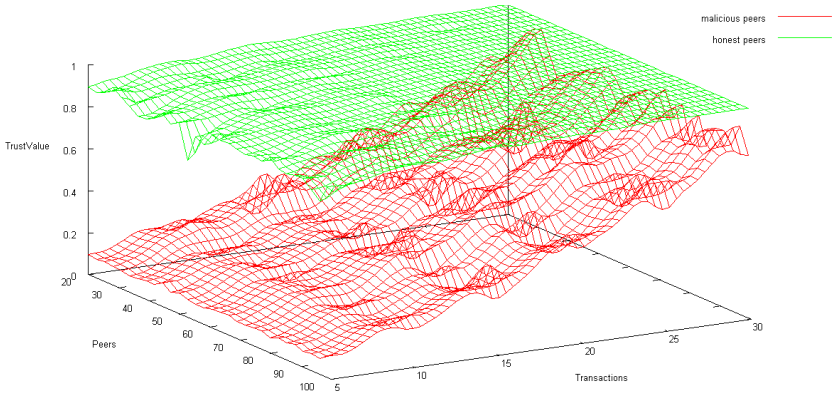

Figure 3.15: Transitive Rating Aggregation in Presence of Ballot Stuffing

3.4.8.3 *Bad-Mouthing in Conjunction with Ballot Stuffing*

The simulation of a bad-mouthing attack differs from the aforementioned simulation, since bad-mouthing needs more coordination between the attackers.

The attackers build a group of malicious peers, the size of the group is defined by a simulation parameter. We also set a certain peer as the initiator. This peer's objective is to benefit from the bad-mouthing attack. We assume that the initiator wants to to drive one of his competitors out of the market. The group has to agree on the victim and the time for acting against the victim. The time is defined by a simulation parameter.

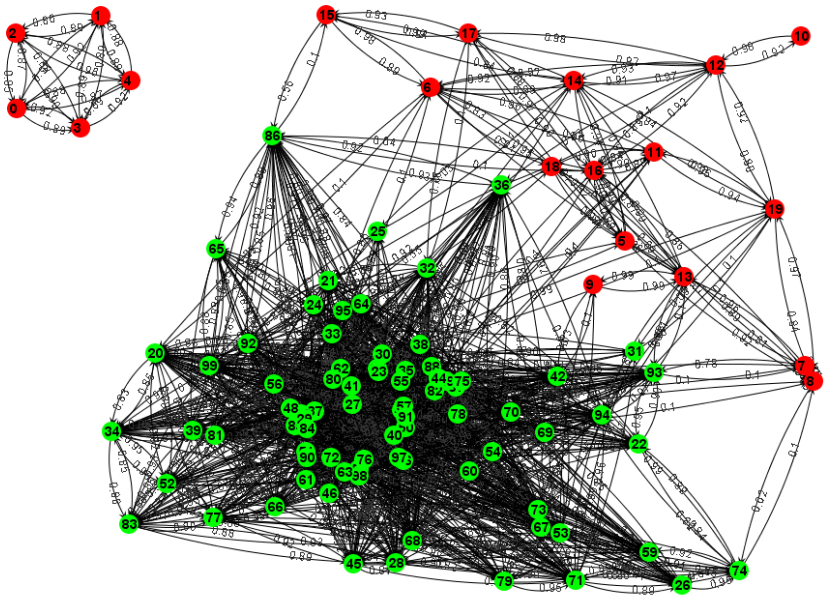

Figure 3.16: P2P Network before Bad-Mouthing Attack

Choosing the victim depends on the situation in the network at the defined time. Since the initiator wants to put a competitor, i.e.,

a peer, out of business by damaging his reputation, the victim's reputation should be better than the initiator's before the attack. After receiving poor ratings from the group of attackers, the reputation of the victim should be worse than the initiator's reputation. The latter remains unchanged, since the initiator does not interact with the victim to avoid receiving a poor rating. Figure 3.16 shows a P2P network before a bad-mouthing attack.

The group of attacker consists of five peers, which have interacted only among each other so far. Before the bad-mouthing attack, the group of attackers has to choose a victim. This is done in consideration of the initiator's trust value. The initiator is peer 0 and his trust value is 0.9007038. Table 1 shows possible victims. Since peer 79's trust value would decrease the most after the attack, the group of attackers chooses this peer as the victim.

Peer Id	Current Trust Value	Anticipated Trust Value after Attack	Anticipated Damage
21	0.9035478	0.7428383	-0.15786551
26	0.9036963	0.7344971	-0.16620667
79	0.9020773	0.7332189	-0.16748486

Table 1: Possible Victims for Bad-Mouthing

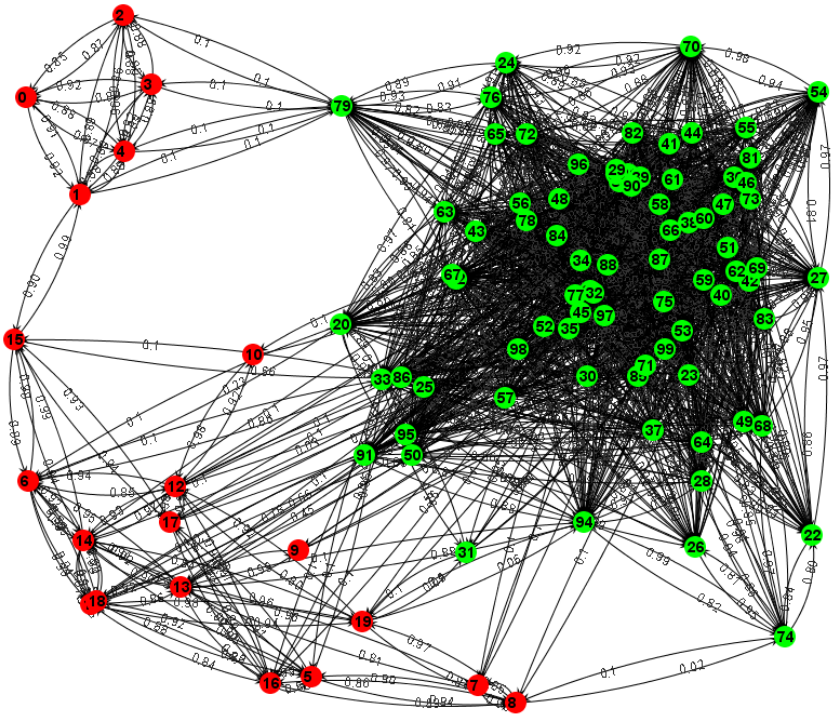

Figure 3.17: P2P Network Immediately after Bad-Mouthing Attack

Figure 3.17 shows the network immediately after the bad-mouthing attack. The group of attackers consisting of peers 1, 2, 3 and 4 have interacted with peer 79 which received poor ratings from the attackers. Peer 0 stays in the background, i.e., he does not interact with peer 79 to avoid a poor rating.

Figure 3.18 shows the network 15 steps after the attack. In this period of time all peers behave as described in the previous section, i.e., malicious peers try to improve their reputation with the help of ballot stuffing and honest peers interact only with peers which have a minimum trust value.

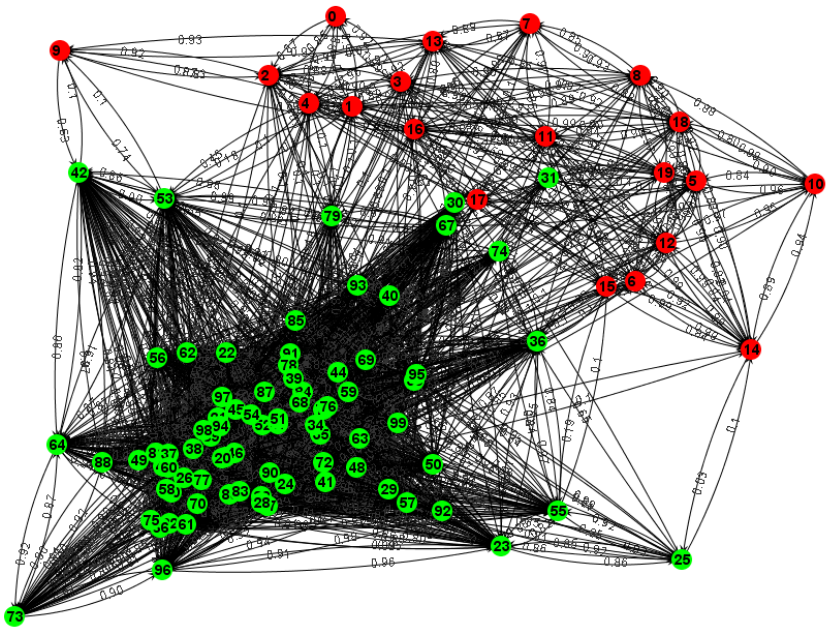

Figure 3.18: P2P Network 15 Steps after Bad-Mouthing Attack

We evaluate the robustness of Algorithm 1 against bad-mouthing. For this, we calculate trust values for the network shown in Figure 3.18. Figure 3.19 shows the resulting trust values. Again, we observe that the calculated trust values converge in course of time.

3.4.8.4 *Robust Transitive Rating Aggregation*

In the previous section, we found out that the average trust value for a malicious peer calculated by Algorithm 1 increases depending on the number of transactions. The reason for this phenomenon is the increasing number of so-called parallel paths. We explain parallel path with an example given in [JHP06] . It is shown in Figure 3.20. It shows three parallel paths from A to E, namely $(A \rightarrow B \rightarrow C \rightarrow E)$, $(A \rightarrow D \rightarrow C \rightarrow E)$ and $(A \rightarrow B \rightarrow D \rightarrow C \rightarrow E)$.

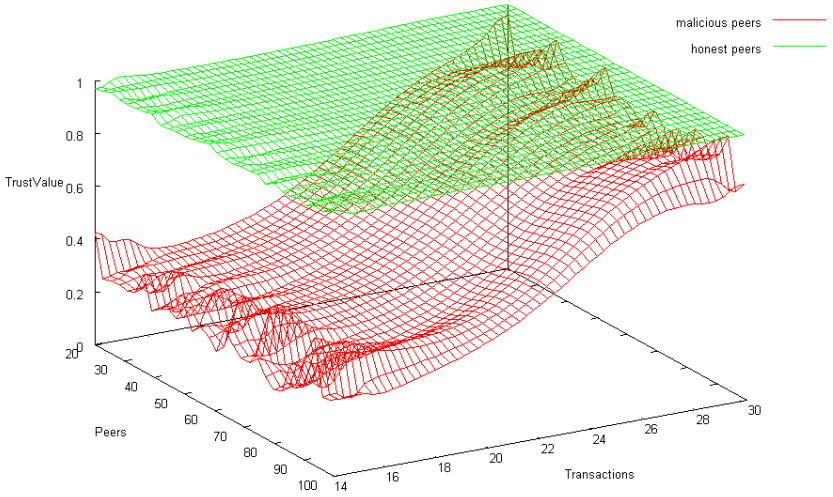

Figure 3.19: Transitive Rating Aggregation in Presence of Bad-Mouthing

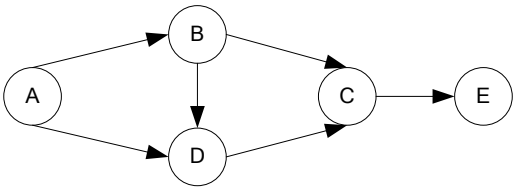

Figure 3.20: Parallel Paths

The weight of an edge is between 0 and 1. Algorithm 1 multiplies the weights of edges to an end vertex for each parallel path, then the result is subtracted from 1. The result is the trust value. It highly depends on the number of parallel paths, i.e., incoming edges. Since malicious peers do not stop to interact with each other, the number of edges directed towards this kind of peers increases with the time. Our simulation shows that the algorithm proposed by Caronni is vulnerable to ballot stuffing and bad-mouthing attacks.

To overcome this shortcoming, we propose an algorithm which does not depend on the number of parallel paths. Instead of mul-

tiplying their weights, we calculate their average. This approach is shown in the following algorithm:

```
   Input: trust network W, start node A, target node Z
   Output: TrustValue
1  i = 0;
2  TrustValue = 0;
3  for all direct neighbors of Z do
4      direct neighbor is X;
5      i++;
6      if X is the start node then
7          TrustValue = TrustValue + edge_{X,Z};
8      else
9          create a copy C of W;
10         drop target node Z in C;
11         returnValue = calculateTrust(C,A,X);
12         TrustValue =
           TrustValue + (returnValue · edge_{X,Z});
13 TrustValue = TrustValue/i;
14 return TrustValue;
```

Algorithm 2: Robust Transitive Rating Aggregation

This algorithm is similar to Algorithm 1. Both of them are recursive but they vary in the way of aggregating parallel paths. Our proposal shown above counts the number of parallel paths in line 5 and returns the average of them in line 14. This value is calculated in lines 6, 12 and 13.

To evaluate the efficiency of our proposal, we calculated the trust values for the P2P network shown in 3.14. Figure 3.21 illustrates the result. The two surfaces remain divergent with increasing steps, thus our proposal is robust against ballot stuffing attacks.

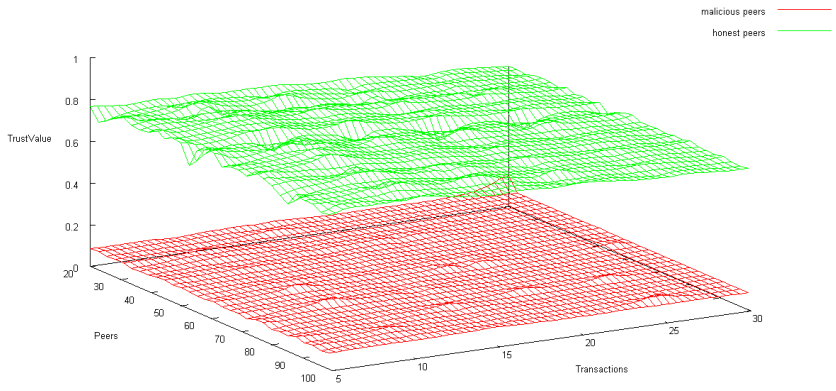

Figure 3.21: Robust Transitive Rating Aggregation in Presence of Ballot Stuffing

We analyze the robustness of our algorithm 2 against bad-mouthing attacks using the network shown in Figure 3.18. The result is depicted in Figure 3.22.

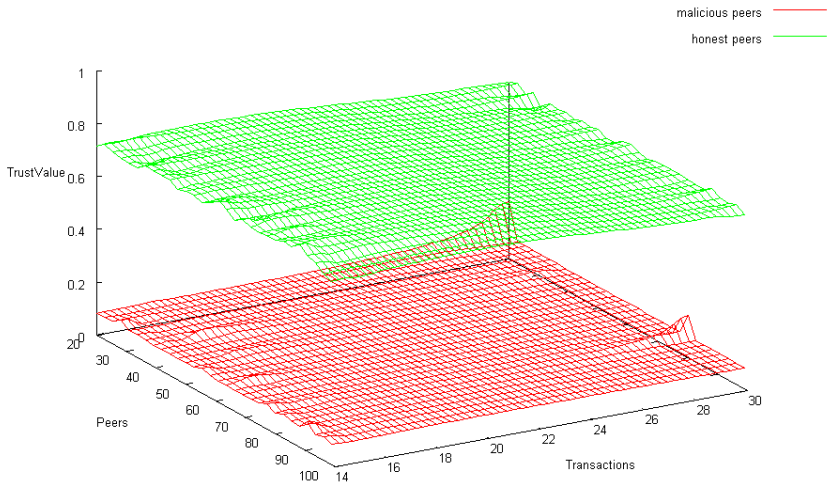

Figure 3.22: Robust Transitive Rating Aggregation in Presence of Bad-Mouthing

The figure shows that the trust values for honest and malicious peers differ strongly from each other over the course of time. In other words, our algorithm for transitive rating aggregation is robust against bad-mouthing.

3.4.9 *Evaluation*

Like numerous other systems, modeling a reputation system for P2P networks is also a trade-off because of conflicting requirements. For example, making a reputation system robust against malicious behavior means implementing additional actions. This again leads to some overhead caused by the reputation system. We evaluate our reputation system with the help of the requirements identified by Zhou and Hwang [ZH06] and discussed in Section 3.2.1 in the following.

HIGH ACCURACY Our model helps to distinguish reputable peers from malicious ones. This is done by the algorithm for transitive rating aggregation which is robust against attacks aiming at manipulating the reputation of peers.

In addition, since our reputation system ensures that the values of two mutual ratings are independent from each other, the resulting reputation is based on the real experiences of the raters.

FAST CONVERGENCE SPEED Since new transaction ratings are published immediately after a concluded transaction, the

(new) reputation information of the involved peers is propagated fast to other peers. In addition, each peer is responsible for the rating aggregation. As a consequence, they will not be any delays caused by central entities.

LOW OVERHEAD Our system causes additional overhead for computation and communication. This overhead is not investigated yet.

However, it seems reasonable that in some cases the use of the reputation system is inappropriate. For example, if two peers have closed numerous joint transactions and have given good ratings to each other, they should not use the reputation system for new transactions because new positive transaction ratings would not improve the reputation of those peers.

ADAPTIVE TO PEER DYNAMICS Our system in only partly adaptive to peer dynamics. Requiring that all involved parties, especially the witnesses, are online during the whole transaction is a strong assumption. However, since our reputation system does not require any predetermined peers, it is adaptive to peer dynamics.

ROBUST TO MALICIOUS PEERS Our reputation system does not prevent malicious behaviors, but makes it difficult. A group of peers can collude to improve the reputation of its members. However, this malicious act requires a big effort. The malicious peers have to follow the steps defined in Sections 3.4.5 - 3.4.7. Nevertheless, this malicious behavior can be detected during the information phase with the help of the algorithm proposed in Section 3.4.8.4.

The robustness of our reputation system is based on the data structures proposed in Section 3.4.4. In particular, the data structures for single transaction ratings and rating lists ensure that manipulations of these data objects will be detected.

SCALABILITY Our approach does not require any central instances. The witnesses are normal peers. Therefore, it can be used by a large number of peers as well as by a small peer group.

3.5 SALE BY CALL FOR TENDERS

We described the business process "Sale by Call for Tenders" (CFT) in Section 2.2.2. A naive translation of the CFT to open

networks introduces new manipulation possibilities like identity masquerading, repudiation of messages etc. We explain these attacks in Section 3.5.1. Then, we show in Section 3.5.2 how the basic CFT can be made robust against these classical security attacks. However, this approach does not eliminate fundamental economic design problems of the CFT itself. In Section 3.5.3, we show how the CFT can be protected against manipulations that damage the fairness and economic efficiency of a market by turning it into a secure sealed-bid auction protocol.

3.5.1 *Requirements for Secure CFT*

The CFT process introduced in Section 2.2.2 works well in a closed environment with a small number of buyers and sellers with stable business relationships. That means the business partners know and trust each other.

In Internet-based marketplaces all these assumptions usually do not hold. Business relationships are much less stable, buyers and sellers can more flexibly join and leave a market, and there are virtually no limits to the number of market participants. A naive adoption of the CFT to open networks is subject to manipulation. We can distinguish four main sources of attacks:

EAVESDROPPING OF MESSAGES Messages may be spied out and the observed information can be misused by unauthorized third parties.

MASQUERADING OF IDENTITY Parties can act under wrong identities.

MESSAGE MANIPULATION A message can be modified during its transmission.

REPUDIATION OF MESSAGES A party sending a message within a CFT process may later claim it has not originated it or it may dispute the exact contents of the message.

From the above attacks we derive requirements for a CFT applied in an open marketplace setting.

DATA CONFIDENTIALITY (PRIVACY) The messages exchanged during the CFT process between seller and bidders should be private. Thus, it should not be possible for an unauthorized third party to eavesdrop the message contents.

MESSAGE ORIGIN AUTHENTICATION The receiver of a message should be assured of the identity of the sender to avoid masquerading of identity.

MESSAGE INTEGRITY An unauthorized party should not be able to modify or corrupt message contents without being detected in order to guard against message manipulation.

TRANSACTION AUTHENTICATION Each message which is exchanged should be unique, such that it cannot be intercepted and replayed by an unauthorized third party without being detected.

NON-REPUDIATION To prevent the subsequent denial of a CFT, all messages exchanged among sellers and bidders need to be legally binding and non-repudiable. Thus, the sender of a message cannot later deny ownership of the message if it was sent with his/her digital signature.

We discuss in Section 3.5.2 how the basic CFT can be supplemented with non-repudiable communications including authentication of origin, message integrity, and transaction authentication. This approach ensures that no new manipulation possibilities are introduced by the realization of CFT with computer support. However, this approach does not eliminate fundamental design problems of the CFT process itself. Further drawbacks of the CFT protocol surface in the economic design of the protocol. The main source of manipulation caused by the design of the CFT are:

ASYMMETRY OF KNOWLEDGE In the bidding phase of a CFT the state of knowledge on the current bidding status is unevenly distributed between the bidders and the seller. During this phase a seller can open the present offers and exploit this knowledge. For example, the seller can inform other market players on the current bidding status and motivate them to outperform the current bids.

ASYMMETRY OF CONTROL The manual winner selection is solely controlled by the seller. In contrast to auction protocols, the CFT thus cannot give any guarantees with respect to the winner selection. In practice, the winner determination is based on criteria beyond the price like a bidder's identity and the trust already established in the business relationship between the bidder and seller.

In addition, the CFT protocol can result in a sub-optimal allocation of goods. The manual winner selection cannot guarantee that the bidder selection is the most efficient solution, i.e., the solution that maximizes the surplus of the economy. Furthermore, the protocol motivates bidders to speculate on the bids of their opponents.

To overcome the manipulation possibilities caused by the asymmetry of knowledge and the asymmetry of control an extended CFT protocol demands the following requirements:

BID CONFIDENTIALITY To avoid the manipulation possibilities caused by the asymmetry of knowledge, we demand that bids can only be opened when the CFT expires. Additionally, it should be ensured that invalid bids cannot violate the integrity of the CFT process.

FAIRNESS To avoid asymmetry of control the winner determination needs more transparency for the bidders to increase their trust in the CFT process. For a fair winner determination, it is required that the rules for the winner determination are stated a priori by the seller in the CFT, e.g., only best price like in auctions or a combination of price and delivery time etc. One prerequisite for fairness is that rules allow to rank the bids in a total order where the highest ranked bid is the winning bid.

In addition, a fair CFT requires that the bidders can restrict the amount of information revealed to the seller. That is, only the information required to determine the winner should be accessible by the seller. Thus, information like identity, location, business profile etc. should only be accessible if required by the seller for the winner determination.

The existence of comprehensible winner determination rules also allows bidders to verify the correctness of the selection process carried out by the seller.

3.5.2 *CFT with Non-Repudiation*

In the following, we will present an electronic version for a reliable CFT that allows the involved parties to detect a variety of attacks performed by malicious parties and provides legal bindingness by using techniques for non-repudiation. The goal of our approach is to cope with problems that result from

- eavesdropping messages,
- manipulating messages,
- masquerading of identities,
- replaying messages,
- repudiation of messages.

The first problem in this list can be solved by encrypting the transmitted data. The encryption of data at the transport level can be achieved via TLS [DR06]. In the following, we assume that all data are transmitted in an encrypted way. In order to tackle the remaining vulnerabilities, there are basically two security concepts that will be applied: digital signatures and availability of authentic public keys. The combination of them ensures that any modification of signed documents can be easily detected since this would cause the invalidity of the signature. Additionally, the identity of the signer can be obtained by the certified public key assigned to a specific identity. Replay of old statements can be detected if the signed documents are unique. Such attacks can be avoided by the usage of sequence numbers or time stamps and a receiver that will never accept an identical message twice. The non-repudiation results from the property that no other party is able to calculate the digital signature since the secret key is known exclusively by the owner.

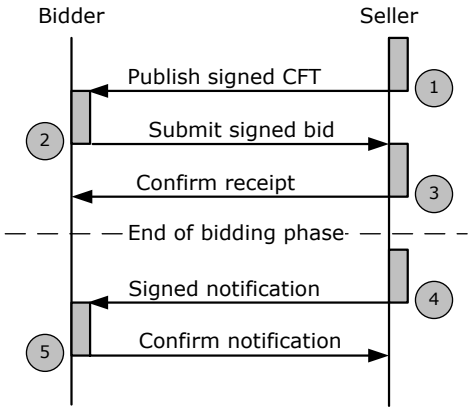

Figure 3.23: The Sale by Call for Tenders Protocol

The security of the electronic CFT is achieved by designing a protocol that basically applies digital signatures relying on a infrastructure that guarantees the authenticity of public keys. Therefore, it is necessary to find a suitable sequence of messages that have to be exchanged between the seller and the bidders in order to meet the previously described requirements. In the following, we will explain the protocol steps which are depicted by Figure 3.23 and give reasons for them.

1. In the first step, the seller issues a digitally signed CFT. It has to be emphasized that the signed message should include the seller's identity, a description of the goods to be sold, and all further relevant conditions of the CFT such as the time in which the seller will accept the submission of bids. Because of the digital signature a bidder can verify

the integrity and the validity of the CFT. Since the signature also depends on some time constraints replay attacks can be avoided. Thus, a malicious party cannot take a copy of an old CFT and publish it elsewhere in order to make a buyer submit his bid. Since the origin of the CFT can be verified the bidder can be sure that this CFT does not come from a fake seller. In case there is one very famous seller in the market that attracts a lot of bidders, other sellers could claim his identity and thereby draw the attention of some customers to themselves. A further motivation for masquerade could be the damage of reputation of an honest seller. All these attacks can be avoided if the seller signs all the information that he publishes to make the authenticity verifiable.

2. The second step deals with the submission of bids. Here, we assume that the bidder takes a form that is provided by the seller and fills in all the relevant data such as his identity and his bid amount and finally signs all this information before submitting it to the seller. With this signed bid the bidder declares that he is willing to buy the offered goods for the price proposed by himself while respecting the seller's conditions. The form provided by the seller should contain the corresponding signed CFT or a reference to it. The application of a digital signature helps the seller to verify the integrity of the bid and to identify the bidder. Furthermore, replay attacks are not possible since there exists a unique relation between the CFT and each specific bid. Thus, if the seller receives a signed message twice this would have no effect as long as the offered goods are only sold to one bidder as a whole. In another context, if bidders are allowed to ask for some smaller portions of the offered goods, and if more than one winner is allowed, then some further bid identification information should be included. If not, then a malicious party is able to send copies of one signed bid in a replay attack and thereby bids for multiple portions of the offered good while masquerading the bidder. The signature on the bid can be used as an evidence by the seller in the case of a dispute in which the bidder denies his bid. Such an evidence can be used to convince any other party that the bidder behaves in a malicious way.

3. In the third step, the seller sends a signed confirmation of receipt to the bidder. Such a confirmation includes a unique reference to the received bid. This confirmation ensures the bidder that his bid will be considered in the determination

of the winner. But this confirmation is also of use for the seller regarding his interests. Without the existence of such a confirmation, a malicious bidder that is powerful enough could try to intercept all concurrent bids. This would be advantageous for him since in the winner determination other bids would be out of the game. By such an attack, the seller does not become aware of better bids. But since in the protocol considered here the bidders expect a confirmation of their bids such an attack would become obvious.

4. After the declared time interval of the bidding phase the seller selects the winner(s) and performs the fourth step in the protocol. This step focuses on the notification of the winner(s). For the sake of simplicity and without constraining generality, we assume that there is one winner. This notification should be signed by the seller for several reasons. Furthermore, the notification should be uniquely related to the bid. The winner can ensure that the message is not faked since he can verify its origin and integrity. Replaying the notification message would make no sense since the receipt of copies will not mislead or confuse the winner. Even a malicious winner cannot gain any profit from copying the notification and requesting the goods for each copy since each acceptable notification is unique and copies will not be accepted. On the other hand, the signature of the notification provides an evidence that can be used in the case of a dispute that may arise if the seller changes his decision after the notification. In such a case, the evidence will entitle the winner to claim the goods.

5. In the fifth step, the winner confirms the receipt of the notification message exchanged in the previous step. In order to make this confirmation undeniable, the winner has to sign it. If such a confirmation is not included in the protocol a malicious seller could leave out the notification in the previous step and claim afterwards that he has sent this notification. Thereby, he would be able to request the amount of money included in the undeniable bid of the second step without informing the winner. In order to have an evidence that he correctly followed the rules of the game the seller has to collect the winner's confirmation. But on the other hand, this confirmation also protects the seller against a malicious winner that denies having received the notification. If a malicious winner refuses to send the confirmation then the seller could force him to do so by using an

official delivery service. Thus, the seller has the guarantee that he will always have the required confirmation.

The CFT protocol as previously described is more or less an adaptation of the real world CFT scenario to the electronic world using security techniques to avoid the described attacks. Thereby, it is the intention of the seller to have a rather unbalanced model. In the CFT protocol it is requested that the winner determination is done manually and also that the rules for this process are not obvious for the bidders. Furthermore, the whole CFT process is controlled exclusively by the seller and he gets a complete insight in all the strategies of the bidders. Thus, besides all the security that was introduced by the previous protocol, there are still various possibilities for unfairness. These problems can be solved by using sealed-bid auction models.

3.5.3 *Sealed-bid Auction with Trusted Third Party*

Auctions provide another approach for price discovery mechanisms and for the selection of suitable partners for business relations. In this section, we present a secured solution for electronic sealed-bid auctions. The auction approach provides an interesting alternative for a CFT since it overcomes some shortcomings of the secure electronic CFT, e.g., concerning fairness. In the CFT model, the seller was free to determine the winner in whatever way he decided. The rules for this process were not public. In contrast, we assume that the rules for the winner determination in auctions are publicly known [KF98b, KF98c, MM87]. Furthermore, all bidders have the possibility to check whether the seller, i.e., the auctioneer, really follows these rules. Thereby, we achieve another quality in this kind of business processes dealing with bid collections and selections from various offers.

The focus of this section is on the sealed-bid auction type. In general, in sealed-bid auctions submitted bids are not visible for any bidder. A sealed-bid auction basically consists of two phases: the bid collection phase and the winner determination phase. During the bid collection phase, the bids are submitted in a way that their contents remain hidden not only to competitive bidders but also to the auctioneer. This prevents an unfair auctioneer from colluding with another bidder by notifying him about the best bid. Before the bids are revealed, the auctioneer commits to the sealed bids. In the winner determination phase, the bids are revealed and the winner is singled out. Our solution thus ensures that all involved parties are able to verify that the auctioneer really follows the auction rules without possibilities to cheat.

Furthermore, there is a need for means that prevent bidders to deny or to withdraw their bids once they have submitted them.

Further properties of our approach for sealed-bid auctions are non-repudiation, anonymity of the bidder, and prevention of message manipulation and masquerading attacks. The auction model approach involves the role of a trusted third party (TTP). The reason for the introduction of a TTP is mainly motivated by the goal of anonymity to be combined with the prevention of bid withdrawals. Anonymity is necessary since the auctioneers could draw conclusions from the knowledge of a bidder's identity concerning his concealed bid. Other work in the context of anonymity can be found, e.g., in [Cha85, Cha88, GRS96, RSG98].

In our concept, the TTP has to be trusted only by the bidder, the auctioneer's trust is not necessary. Thus, there arises no difficulty in finding a TTP which is trusted by both parties. Each bidder can choose any desired TTP. The auctioneer can be in contact with several TTPs executing the protocol. Besides the reason of anonymity, the existence of a TTP offers a higher level of comfort for the bidders. By using a TTP's service they do not have to care about sending messages in time according to the time conditions defined in the auction rules.

In contrast to the traditional sealed-bid auction [MM87], in our model the auctioneer gets insight into all bids during the winner determination phase. This decision is motivated by the fact that under certain circumstances it is of high interest to the auctioneer to find out about the demand for specific goods. This knowledge can be used in organizing subsequent auctions. A further property of our approach is using efficient primitives that results in feasibility.

For the sake of simplicity, we will assume in the following that the winner determination rules concentrate exclusively on the bid amount. In a more general approach, there could also be further variables besides the bid amount that could affect the winner determination, provided that they can be put in a total ranking. However, by our restriction we do not constrain the generality of our approach.

We assume that the bidder has somehow received all the relevant information about the auction, e.g., the goods that can be purchased in the auction, and the auction rules like the auction start time and deadline. For reasons of authentication, all this information is signed by auctioneer A. Before bidder B_i participates in the auction he selects one of the available TTPs to use its services. Another bidder B_j also interested in the auction can select either the same or a different TTP. During the bid collection phase, B_i starts the following protocol:

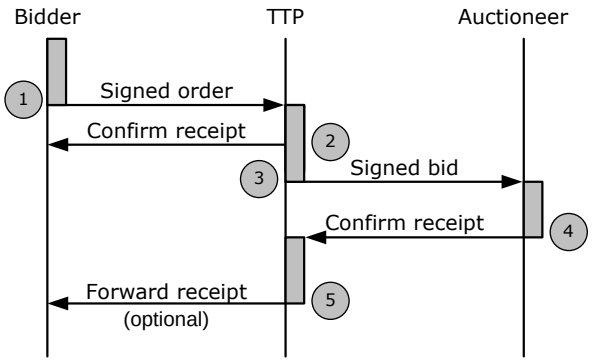

Figure 3.24: Protocol for the Bidding Phase

1. In the first step of the protocol, B_i prepares an order $order$ that consists of the following parts to be sent to the TTP:

$$\begin{aligned} order_i \;=\; & order\_details_i,\ Enc(K_i, amount_i), \\ & Enc(PK_A, K_i), Sig(SK_{B_i}, (order\_details_i, \\ & Enc(K_i, amount_i),\ Enc(PK_A, K_i))) \end{aligned}$$

   In this context, these variables have the following definitions. The variable $order\_details_i$ describes all necessary information that is needed later by the TTP in order to participate in the auction on behalf of B_i such as identity of B_i, the auction B_i is interested in, the auctioneer A, and auction details (e.g., rules, time constraints). The term $Enc(K_i, amount_i)$ describes the ciphertext obtained by symmetric encryption of $amount_i$, i.e., the amount of money B_i is willing to pay in the auction, using key K_i which is randomly chosen by B_i. This ciphertext will be used later as a part in the sealed bid to be forwarded to A. As long as A does not possess K_i he is not able to reveal the amount. The term $Enc(PK_A, K_i)$ specifies the ciphertext obtained by asymmetric encryption of K_i using the auctioneer's public key PK_A. Using asymmetric encryption, a secure channel from B_i to the auctioneer via TTP for later exchange of K_i is established. B_i's digital signature on a document doc is described by $Sig(SK_{B_i}, doc)$. This part fulfills the requirements concerning message integrity, data origin authentication and non-repudiation.

2. In the second step, TTP confirms the receipt of the message obtained in the previous step. In order to do this, TTP replies a digital signature on the received message.

3. In the next step, TTP extracts some parts of the message obtained in the first step and creates a bid to be forwarded to A. This bid contains the following elements:

$$\begin{aligned} bid_i \;=\; & auction\_details,\ Enc(K_i, amount_i), bidID_i, \\ & Sig(SK_{TTP}, \\ & (auction\_details,\ Enc(K_i, amount_i), bidID_i)) \end{aligned}$$

   With bid_i, auctioneer A possesses a commitment concerning the amount of money B_i is willing to pay without knowing the amount. This means that neither B_i nor TTP are able to change this amount afterwards. Furthermore, signing the bid by TTP meets the requirement of non-repudiation. In addition, B_i's identity remains unknown to the auctioneer. Thereby, the requirement of anonymity remains fulfilled. The $bidID_i$ is selected by the TTP and will be used afterwards for an easy re-identification of bid_i. Thus, the $bidID_i$ has to be unique.

4. In this step, the auctioneer informs TTP that he has included the received bid in the actual auction by replying a signed declaration. Thereby, TTP is ensured that bid_i was received and will also be considered in the auction. Furthermore, TTP and therewith B_i have an evidence in a possible case of dispute that bid_i has to be considered in the winner determination phase.

5. TTP passes the received confirmation to B_i after having signed it again. Thereby, the latter knows that TTP has executed his order properly. This step is not mandatory. Since B_i trusts TTP this message serves more the purpose of notification than that of evidence.

Let us assume that the auctioneer receives n bids via m TTPs with $1 \leqslant m \leqslant n$ during the bid collection phase. After the end of the bid collection phase, the winner determination phase which is shown in Figure 3.25 starts. In this phase, the involved parties continue with the following protocol:

6. After having collected n bids via m TTPs, the auctioneer composes a new message list which contains the $auction\_details$ referencing to the corresponding auction, all the received ciphertexts of the bid amounts $amount_1, \ldots,$

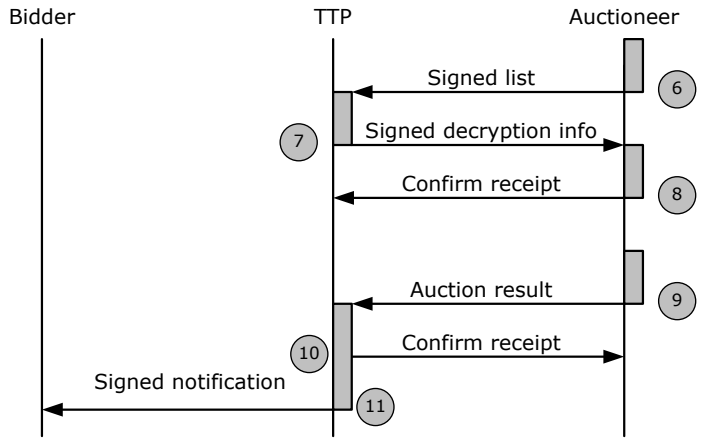

Figure 3.25: Protocol for the Winner Determination Phase

$amount_n$, the bid identifications $bidID_1, \ldots, bidID_n$, and his signature on all these variables.

$$\begin{aligned} list \;=\; & auction\_details, \\ & Enc(K_1, amount_1), bidID_1, \ldots, \\ & Enc(K_n, amount_n), bidID_n, \\ & Sig((SK_A, (auction\_details, \\ & Enc(K_1, amount_1), bidID_1, \ldots, \\ & Enc(K_n, amount_n), bidID_n)) \end{aligned}$$

By this message, the auctioneer creates a commitment on all the encrypted bids he received. After the generation of this message, he distributes it to all the TTPs involved in the auction. Thereby, each TTP is able to verify whether all the bids submitted by himself are taken into account in the auction. If not all the bids are contained in $list$ the TTP can complain using the evidence he received in protocol step 4. The same evidence can also be used in the case if one TTP does not receive $list$ in time. The time constraints are known to all participants by the auction rules.

7. Upon receipt and positive verification of $list$, all involved TTP_i for $i = 1, \ldots, m$ generate a new message $resolve\_info$ containing, beside others, the secured decryption keys for all the encrypted bids with identities $bidID_{i_1}, \ldots, bidID_{i_l}$ sent by the same TTP_i where $\{i_1, \ldots, i_l\} \subset \{1, \ldots, n\}$. Fur-

thermore, the message contains the TTP's signature on this content.

$$
\begin{aligned}
\mathit{resolve\_info} \;=\; & \mathit{auction\_details},\\
& \mathit{Enc}(PK_A, K_{i_1}), \mathit{bidID}_{i_1}, \ldots,\\
& \mathit{Enc}(PK_A, K_{i_l}), \mathit{bidID}_{i_l},\\
& \mathit{Sig}(SK_{TTP_i}, (\mathit{auction\_details},\\
& \mathit{Enc}(PK_A, K_{i_1}), \mathit{bidID}_{i_1}, \ldots,\\
& \mathit{Enc}(PK_A, K_{i_l}), \mathit{bidID}_{i_l}))
\end{aligned}
$$

In the case in which a TTP does not send his message in time the auctioneer can request the corresponding *resolve_info* using the evidence of step 3.

8. Upon receipt of the message *resolve_info*, the auctioneer confirms the receipt by signing and returning it. This message assures the TTP that the auctioneer has really received the correct information.

After the auctioneer has collected all the messages *resolve_info* from the m different TTPs he can start the auction resolution in order to determine the winner, i.e., to find the highest amount. In the case of two or more coinciding bid amounts, there can be some further rules to be applied for resolving the auction. For the sake of simplicity and without restricting generality, we assume that there is a unique highest bid amount. In order to execute the winner determination, the auctioneer applies the key K_i to $Enc(K_i, amount_i)$ for $i = 1, \ldots, n$. K_i is obtained by decrypting of $Enc(PK_A, K_i)$ with the auctioneer's secret key SK_A. With this information, the auctioneer is able to compare the amounts, which leads to the determination of the winner in the auction without the possibility to manipulate the result of the auction. He composes a message *result* in which the winning bid, i.e., bid_i, with $amount_{max} = amount_i$ is published besides the remaining $n-1$ bids. With revealing this information, all further parties involved in the auction are able to verify that the auctioneer executed the auction correctly. Since the auctioneer has given a commitment on the encrypted bids in step 6 there is no way to introduce later new bids of other bidders or to manipulate the amount of existing bids. This reasoning is based on the assumption that it is not possible for an auctioneer with a colluding bidder B_j to find a key $\tilde{K}$ that decrypts the committed value $Enc(K_j, amount_j)$ to the decryption result *amount* that is greater than $amount_{max}$. In general, this can be achieved by the introduction of some redundancy in the representation of $amount_i$ for $i = 1, \ldots, n$.

9. The auctioneer composes a message $result$ to be sent to all TTPs participating in the auction. By this message, all TTPs can verify the correctness of the winner determination process.

$$
\begin{aligned}
result \;=\; & auction\_details, \\
& K_1, amount_1, bidID_1, \\
& \vdots \\
& K_i, amount_i, bidID_i, \;\longleftarrow \text{winner} \\
& \vdots \\
& K_n, amount_n, bidID_n, \\
& Sig(SK_{Auc}, (auction\_details, \\
& K_1, amount_1, bidID_1, \ldots, \\
& K_n, amount_n, bidID_n))
\end{aligned}
$$

   In the verification, a TTP checks if all the parameter values sent by himself are properly contained in the message $result$. Additionally, they can apply the keys K_i to their corresponding ciphertexts $Enc(K_i, amount_i)$ for $i = 1, \ldots, n$ to obtain the values of $amount_i$. By the $bidID_i$, the verifiers are able to identify the bids sent by themselves. Furthermore, all TTPs are able to reconstruct the winner determination. If the auctioneer tried to manipulate the winner determination, this can be detected.

10. In this step, the TTP confirms that he received the message $result$ and that he accepts the result of the auction by sending a signed reply.

11. In the last step of the protocol, the TTP informs the bidder about the result of the auction. The message to be exchanged depends on the fact whether the receiving bidder won the auction or not.

In all the steps that were performed during the auction the identities of the bidders remain hidden to the auctioneer. Thereby, we have fulfilled the requirement of anonymity at the same level at which this is also fulfilled in real world auctions. Therein, agents can operate on behalf of their clients. The subsequent steps such as shipment and payment are outside the auction itself. But even there, depending on the services offered by the TTP, these steps can be performed via the TTP. Thus, the winner can remain anonymous to the auctioneer also after the auction is finished.

The protocol was designed to keep the overhead for computation, communication, and the number of required components, e.g., servers, small. Furthermore, the presented auction approach, which focuses on a single-round sealed-bid auction, can easily be adapted to the need of multi-round auctions.

3.6 RELATED WORK

In the following, we review related work proposed by the scientific community and compare them with our contributions in this section.

Repuation Systems

[ZMM99] proposes two complementary reputation systems, namely Sporas and Histos. Sporas is an evolved version of existing online reputation models, e.g., eBay's rating system. In this model, only the most recent rating between two users is considered. Another important characteristic is that users with very high reputation values experience much smaller rating changes after each update than users with a low reputation. In order to prevent a user from choosing a new identity, a user's reputation value will never fall below a beginner's value. Histos is a complement to Sporas and builds personalized system. Similar to our reputation system, Histos is based on the concept of Web of Trust to calculate personalized reputation values. In contrast to our work, Sporas and Histos do not provide any means for exchanging mutual ratings in a secure way. In addition, protecting ratings is another open issue.

[LZRD04] proposes a reputation system, i.e., R-Chain, for P2P networks. It has two similarities to our proposal. First, it uses hash chains to protect rating data from being manipulated. Second, it relies on a witness, which has to be involved in a transaction. However, in contrast to our work, R-Chain does not provide any algorithms for a rating aggregation.

Au et al. present a framework for enabling cross-organizational trust establishment [ALA01]. They introduce trust tokens which are issued by trust servers. These tokens are required when a user wants to access a protected resource. Au et al. also use the concept of Web of Trust to model the trust relationship between the trust servers. However, in contrast to their work, we build up a Web of Trust based on ratings and can calculate the trust level in an ad-hoc manner.

Kamvar et al. propose the EigenTrust algorithm for reputation management in P2P networks [KSGM03]. This work is close to

our system, since both are based on the concept of transitive trust. But in contrast to our work, Eigentrust relies on good choice of some pretrusted peers, which are supposed to be trusted by all peers.

In [Mau96], Maurer proposes an approach to model a user's view of a public key infrastructure (PKI). From this view, a user draws conclusions about the authenticity of other entities' public keys and possibly about the trustworthiness of other entities. A user's view consists of the user's statements about the authenticity of public keys and the trustworthiness of their owners, as well as a collection of certificates and recommendations obtained or retrieved from the PKI. The approach is similar to our work, since it relies on transitive relationships. However, we use other data, i.e., transaction ratings, to calculate the trustworthiness of market participants.

Gupta et al. propose in [GJA03] a reputation system for P2P networks to assess peers' levels of participation in a P2P system. The main goal of this system is to give the peers an incentive to actively participate in the system and to discredit freeriders. For this purpose, they introduce a reputation score which is based on the behavior of the peers as well as on their capabilities. The part of the reputation score that represents the behavior consists of the "content search contribution", i.e., the willingness to forward and process requests, and the "content download contribution", i.e., the willingness to serve data. The second part of the reputation score, which represents the capabilities, consists of the processing power, bandwidth, storage capacity and memory of the peer. Two schemes are used to calculate reputation scores with the help of a centralized "reputation computing agent" (RCA). This approach differs from our decentralized reputation system. In addition, the proposal of Gupta et al. does not enable peers to rate each other after a conducted transaction.

Fair Auctions

In general, the World Wide Web provides a ubiquitous platform for the execution of electronic auctions. In [Kle00], a framework of constituting elements of auctions is presented and the impact of the Web on the proliferation of auctions is discussed. A short discussion on auctions' benefits and a description of the goods and services traded in auctions is given in [Tur97]. Franklin and Reiter present a solution in [FR96] which is based on a distributed system approach in which the security requirements are fulfilled as long as not too many out of the auction servers' set operate maliciously and do not collude. In our context, the requirement

for multiple servers is not suitable for small companies intending to take on the role of an auctioneer. Furthermore in order to prevent collusions between these servers they have to be offered and maintained by different parties.

In [HKT98], an adaption of the first price and second price sealed-bid auctions to computational environments is presented. Like in [FR96], the privacy of submitted bids is preserved by using a form of secure distributed computation. The mechanism ensures that the auctioneers and participants (except for the winner) will be completely unaware of the non-winning bids.

The solution of [KHT98] considers multiple auctioneer servers and multi-round auctions. [KF98a] describes a variety of commonly used auction mechanisms and present a software architecture for electronic auctions. Other work in the area of electronic auctions with focus on security aspects was done in [NPS99, Cac99, SS99, SA99]. In [WTL00], a method to earn strategies for multilateral negotiations such as auctions is presented.

The economic design of auction protocols has been widely discussed in auction literature. The objective is to design auctions on solid economic principles and to ensure that participants have incentives to bid as they truly value the item. An overview and analysis of the underlying economic principles is given in [MM87]. In [Vic61], Vickrey has designed a sealed-bid second price auction that is incentive compatible, and maximizes consumer surplus. The analysis and investigation of the economic properties of auctions usually assumes a trustworthy auctioneer or auction house. With the application of auctions to the Internet this assumption can no longer be maintained. The application of auctions to computational environments and its implications are discussed in [KF98c, Tyg98]. In [San96], problems and limitations of automated Vickrey auctions are discussed and means to circumvent them are developed.

3.7 CONCLUSIONS

In this chapter, we analyzed less organized business processes, in which involved parties, e.g., market participants, have their state of knowledge by having more or better information than others as well as their own motives. As a result, the mentioned parties may distrust each other. In order to overcome this shortcoming, we focused on trust building for open networks. We proposed two different reputation systems and a fair auction protocol. The first reputation system is centralized, i.e., it relies on a trusted third party. This reputation system has two key features. First, it supports market participants during all phases of a market

transaction including the negotiation. Second, it provides a set of different algorithms for rating aggregation which in turn increase the usability of the reputation system. The main features of the second reputation system are its fully decentralized character and its robustness against coalition attacks. As a consequence, the proposed reputation system can be used in P2P networks. We verified the robustness of our decentralized reputation system with the help of a simulation. In Section 3.5, we showed how a concrete business process, i.e., Sale by Call for Tenders, can be made robust against classical security attacks, such as identity masquerading and repudiation of messages. In addition, we explain how to overcome economic design problems of the mentioned business process.

All the proposed approaches are soft security mechanisms and can be applied in business process where involved parties have less stable relationships with each other. In the next chapter, we focus on more organized business processes for which we propose a set of hard security mechanisms for usage control.

4

USAGE CONTROL

This section deals with usage control for electronic business processes. In particular, we focus on controlling the usage of digital objects, e.g, electronic documents, which demands two functionalities. First, we need methods for defining usage rules. Second, mechanisms for enforcing the defined usage rules are required. In this section, we address both aspects of usage control.

Electronic documents play a central role in business processes, since they are a means of integration and are handled among business partners. Some documents are sensitive and thus have to be protected from being accessed by unauthorized parties. For this purpose, we propose in Section 4.1 a flexible access control model for electronic documents. Our model captures the information about the operations performed on documents. This history information can be used to define access control rules. The proposed access control model is the result of our joint work with Patrick Röder and is presented in his PhD thesis [Rö8]. In Section 4.1.7, we show how to apply our access control model in context of business processes.

In Section 4.2, we consider scenarios, in which customers are involved in the execution of a special kind of business processes, such as selling and consuming digital goods. In these cases, digital goods have to be protected from being used in an unauthorized way, i.e., being shared in public networks. This requirement reflects another facet of usage control. In this context, we have to define usage rules for digital content, e.g., music files. These rules have to be enforced on customers' platforms. Thus, the integrity of customers' platform will be verified before transferring digital goods. For this, we propose a robust integrity reporting protocol which is necessary when a remote platform, e.g., a customer's computer, has to perform security relevant operations, e.g., to enforce a security policy. This protocol can be used for developing a robust Digital Rights Management (DRM) system that supports the transfer of digital goods to other users or devices. We present such a DRM system in Section 4.2.7.

[SR01] explains the key advantages of the eXtensible Markup Language (XML). These are the support for multiple views of the same content, selective (field-sensitive) queries over networks, e.g., the Internet, and a standard data and document interchange infrastructure. Consequently, XML is the key technology for storing and sharing of business data. Since some of this data can embody confidential information, protecting this kind of data is crucial. In this section, we propose a flexible model for controlling access to XML documents. We present a usage scenario which is used to drive requirements for access control to XML documents and to explain our model with the help of concrete example rules.

In our scenario, we consider three companies. The first company is SmartChips, which is a supplier of car computers. The other two companies are EcoCars and FastCars, both of them are car manufacturers and competitors as well. Each company has its own standard for the classification of its products or its bill of materials. There are also public standards that can be accessed by all companies without restrictions. Competitors should not be able to access the standards of their adversaries. SmartChips and EcoCars have a strong collaboration. Therefore, SmartChips is allowed to edit the standard of EcoCars and can copy parts of its own standard to EcoCars' standard. Since SmartChips is also a supplier to FastCars, SmartChips has read access to the FastCars standard and can copy parts of this to its own standard. Figure 4.1 shows the permitted data flows in our scenario. In our

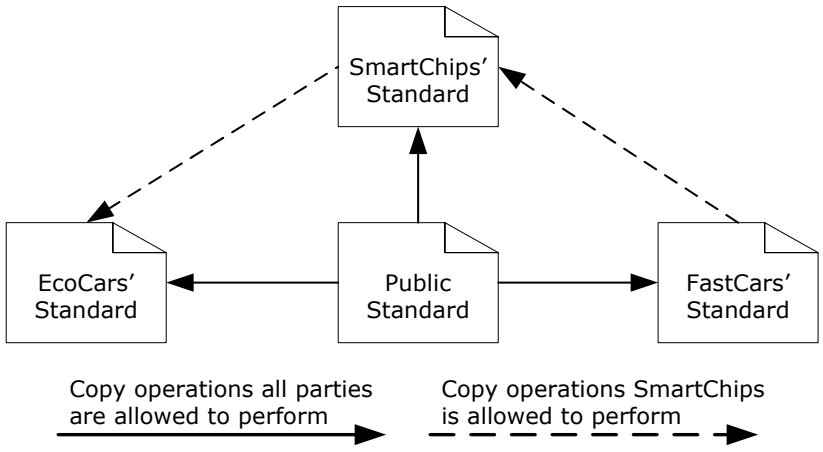

Figure 4.1: Permitted Data Flows

example, we consider a business process between SmartChips and FastCars, during which SmartChips views the standard of

FastCars and copies some parts of this to its own standard. From FastCars' point of view, SmartChips should not be able to copy parts of FastCars' standard to EcoCars' standard to avoid undesirable information flows. For this, the concept of noninterference has been proposed by [GM82]. [Man03] presents the following common intuitive understanding of noninterference: "a group of processes is noninterfering with another group of processes if the actions of the first group have no effect on the observations of the second group." To ensure noninterference, SmartChips' access to EcoCars' standard must be restricted once FastCars' standard has been accessed by SmarChips. Depending on the requirements of the scenario, we can define three alternatives for limiting the access of SmartChips. In the first instance, SmartChips is not permitted to access the standard of EcoCars. In the second, SmartChips is not allowed to edit the standard of EcoCars. In the third, SmartChips is allowed to edit the standard of EcoCars, as long as no data is transferred from FastCars' standard. This case occurs when SmartChips transfers data from a public standard to EcoCars' standard.

This scenario illustrates several requirements with regard to access control. First, we need a mechanism for recording accesses to standards, which are stored in documents. This recorded information must contain information about each operation performed and its context. The context must include relevant data for access decisions, e.g., the subject that performed the operation and the time at which the operation was performed. Second, we must be able to define access to documents depending on former accesses. As a consequence, we need a model where access is defined based on the history of previous operations. Third, since standards contain descriptive texts which can be partly reused in other standards, the access-control system has to be able to define access to these parts individually. Fourth, since standards are usually stored in XML documents, we need a model that can define access to XML documents.

In the next section, we introduce a model that fulfills the four requirements mentioned above. We start with a description of the histories, continue with the operations defined in our model and finally present the syntax of our access rules. Then, we present a system architecture including algorithms which are necessary when we apply our model in a distributed environment. At the end of this section, we show how to use our model to define access for the aforementioned scenario.

4.1.1 *Overview of History-Based Access Control*

Our access control model defines which subjects are allowed or denied to access certain parts of an XML document. Concerning the subjects of our model, we design our model for human users. The objects of our model are different parts of an XML document. These parts can either be entire XML elements (including attributes and text content), attributes or parts of the text content of an XML element. We define a set of operations, which enables the user to view and edit XML documents. The effects of these operations are recorded in the history. When we record an operation in the history, we also log the context information of that operation. Generally speaking, context information can be any information that helps to specify the situation of the operation more precisely. In our case, we record the date and time of the operation, the subject that performed the operation and the role the corresponding subject was active in. Summing up, the history stores how a document was created.

Finally, we use access control rules to define that subjects in a certain role are allowed or denied to perform a given operation on specific objects. These objects are described by a condition, which defines predicates on the content of the current document and on the history. When a user tries to perform an operation on an object, we check whether there is an access control role that matches with the active role of the user, with the operation that he wants to perform and with the object. The first two aspects can be verified directly using the current role of the current user and the invoked operation. In contrast to this, to find out whether a rule matches with an object, we must check whether the object has the properties described by the condition of the rule. Therefore the condition must be evaluated for the current object.

4.1.2 *Histories*

We use histories to keep track of changes caused by the operations `create`, `copy`, `delete`, and `change attribute`. The operation `view` is also logged in the histories. These operations are described in detail in Section 4.1.3. We keep one history for every element itself including its attributes and one history for its text content. The latter history uses markup elements to divide the text into text blocks with a block ID. This mechanism enables us to keep track of sub-elements of arbitrary size. The markup elements are defined by `ac:block` elements, where `ac` is the prefix for the namespace of the access control system. We use the block IDs to reference individual text blocks in the history for the text content.

If a view is created for a user, the `ac:block` elements are omitted. Keeping track of such implicitly defined sub-elements allows us to manage protection units smaller than an XML element. Technically, we use XML elements to define those sub-elements, but from a user's point of view, these sub-elements are not visible.

A new text block is created in two cases. First, we create a new text block as a result of a copy operation, at both the source and the destination element. Second, we create a new text block whenever text is added to an element.

In addition to the histories, we maintain a unique element ID for each element to reference it independently of its current position within the document. Moreover, each document has a unique document ID. We use these IDs to keep track of copy operations by maintaining an *is-copy-of* relation among the elements and text blocks. Two objects are in is-copy-of relation with each other if one object is a copy of the other.

Figure 4.2 shows a graphical illustration of the is-copy-of relation of the objects mentioned in the scenario described in Section 4.1. We refer to this kind of illustration as a copy-graph. In this example, all objects are text blocks. In the general case, objects can be XML elements as well.

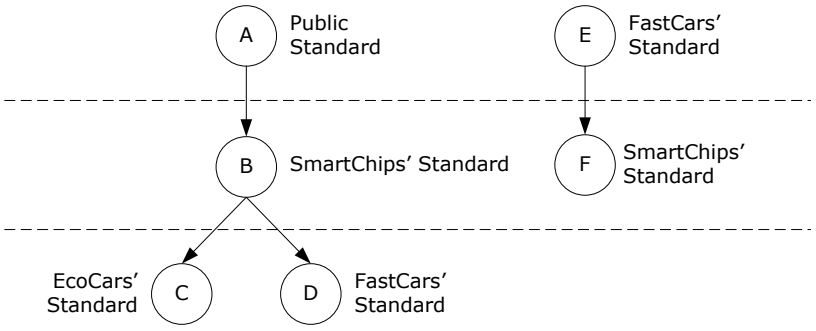

Figure 4.2: Example Copy-Graph

A history entry consists of an action element, which contains details on the operation and a context description. In addition to the operation, an action element can have up to two arguments that describe the action. For the actions related to attributes, we store the name of the corresponding attribute. The `change attribute` and `create attribute` operations additionally store the new value of the attribute. The `create text` and `delete text` operations store the block ID of the corresponding text block.

4.1.3 *Operations*

In this section, we describe the details of the operations supported by our model. These are `view`, `create`, `delete`, `change attribute` and `copy`. Most of the operations can be applied to elements, text and attributes. Each operation, except for `view`, has an effect on the document itself as well as on the histories. The `view` operation creates a history entry only. The `create` operation is divided into creating elements, creating attributes and creating text.

The `create element` operation creates an element without any attributes or text. In addition to the element itself, the history of the element is created. The first entry of the history for the element describes its creation. The attributes of an element are created with the `create attribute` operation, which is also logged with an entry in the history of the enclosing element. It can be required that elements have mandatory attributes. This requirement should be checked on the application level and not within the access control system. This also applies to the deletion of mandatory attributes.

The `create text` operation is used to add new text to an element. This operation has an argument that specifies the position of the new text. If this position is within an existing block, this block is split at the position where the new content should be placed and the new content is placed in-between the split blocks. The split blocks keep their original histories, whereas the new content gets a new history with one entry describing its creation. The boundaries of the split content pieces are denoted by the `ac:block` elements, as described in Section 4.1.2.

The `delete` operation is used to delete elements, attributes, text or parts of the text. Since elements and their attributes are checked in rules, we need to keep them after deletion. For that purpose, the context of a delete operation is captured in the element history with a delete action entry. A context is a tuple of `Date`, `Subject` and `Role`, where `Date` refers to a date including time and `Role` is the role of the subject that performs the corresponding operation.

The `copy` operation is used for elements, text or parts of the text. In all cases, we apply the corresponding `create` operation to create a new instance at the destination as a copy from the source, which is registered in the destination element. Additionally, the is-copy-of relation of the elements is updated.

The `view` operation displays elements which have not been deleted. When a user wants to view a document, the `view` operation is invoked for every element of the document itself, but also for its attributes and text. In contrast to the read operation of

some other systems, e.g., [BL73, BN89], the view operation does not imply a data transfer.

The `change attribute` operation allows users to change the value of a specific attribute. Since former values of an attribute can be checked by rules, we record the change with an entry in the corresponding element history.

4.1.4 *Rules*

In this section, we define a syntax for Access Control (AC) rules, which can express policies that depend on the content of the current document, the recorded history information and the content of dependent documents. Generally speaking, a rule has the typical structure of subject, operation, object and mode. The `mode` field of a rule defines whether it is positive (allow) or negative (deny). The default semantics of our model is deny: if the access to the object is not defined by a rule, then the object is not accessible. If conflicts occur, we take the rule with the superior role and finally apply "deny takes precedence over allow".

We use roles [SCFY96] to model the subjects to gain a higher level of abstraction and therefore more flexibility compared to directly listing individual subjects.

Instead of listing individual objects in rules in an ACL-like manner [GD72], we describe objects by their properties, e.g., location within a document or attribute values. For this purpose, we use XPath patterns [BBC+07] to describe the objects for which a rule is applicable. We use XPath, since its clearly defined semantics presented in [DFF+07] makes the interpretation of the resulting rules unambiguous. Moreover, XPath has a predefined set of mechanisms that can be used for our purpose.

We define two types of rules. The first type of rules defines permissions for the unary operations `create`, `view`, `delete` and `change attribute`. The objects of an AC rule are defined by an XPath pattern. The second type of rules defines permissions for the binary `copy` operation, which requires the specification of a source and a destination object. We use two XPath patterns for this. The syntax of both types of rules is listed in Figure 4.3.

4.1.5 *Accessing History Information with XPath*

We use XPath patterns in rules to define access depending on histories. As a consequence, we need a mechanism to access the histories within an XPath pattern. Therefore, we extend the function library of XPath by a set of functions, which we collect in the following six groups. The namespace of our functions is

Unary rule		*Copy rule*	
Element	Description	Element	Description
Role	role	Role	role
Operation	view|create|delete| change attribute	Operation	copy
Object	XPath	Object	XPath
		Destination	XPath
Mode	allow|deny	Mode	allow|deny

Figure 4.3: Syntax of Access Control Rules

indicated by the prefix 'ac:'. In the context of XPath, we speak of a node instead of an object.

Getting Copies of a Node

This group of functions is related to the is-copy-of relation of nodes among each other. It is required to express rules that define access depending on the source of an object or on the locations to where an object was copied.

The function `ac:copies` returns all nodes that are in is-copy-of relation with the current node, whereas the function `ac:predecessors` returns all nodes of which the current node is a copy. Finally, the function `ac:successors` returns all nodes that are copies of the current node. All three functions also return nodes that are in indirect is-copy-of relation to the current node, e.g., `ac:successors` also returns the copies of the copies of the current node.

Getting Attribute Values

The function `ac:attribute-values` returns a chronologically sorted list of tuples of an attribute value and the context corresponding to the change of the attribute value. It is required to define rules, which inspect former values of an attribute. For example, the rule {`researcherB`, `View`, `deny`, `/Report[count(ac:attribute-values('funded-by')[value='Company A']) > 0]/*`} states that subjects in the role `researcherB` are not allowed to view reports that were funded by `'Company A'` in the past or at present.

Getting Related Nodes Depending on Time

This group of functions retrieves nodes addressed relatively to the context node that existed within a specified time interval. In the XPath terminology, the element to be checked against the pattern is called the *context node*. XPath offers functions to retrieve nodes addressed relatively to the context node, but without the specification of a time interval, since XPath only considers the current state of a document. This time interval is required to select related nodes depending on time, since nodes can be deleted. Therefore, each of these functions can have a time interval as parameter, e.g., `ac:children-at(t1, t2)` returns all nodes that were children of the context node in the time interval between `t1` and `t2`. To inspect a single point in time, `t2` can be omitted. The functions of this group are `ac:parent-at`, `ac:following-at`, `ac:preceding-sibling-at`, `ac:preceding-at`, `ac:following-sibling-at`, `ac:children-at`, `ac:descendant-at`, `ac:root-at` and `ac:self-at`.

Getting the Context of a History Entry

This group of functions offers access to the context of a specific history entry. Each function returns an element consisting of subject, role and time. These functions are `ac:creation-context` and `ac:deletion-context`.

Getting Accessed Nodes

This group of functions is used to get all nodes which have been accessed by a specified user or by a user in a certain role. Amongst others, these functions are required to express Chinese Wall policies [BN89]. The functions are `ac:created`, `ac:viewed`, `ac:changed-attribute` and `ac:deleted`. Each function refers to a specific operation, e.g., `ac:viewed` returns viewed nodes. In addition, the function `ac:accessed` returns all accessed nodes independently of the operation. All functions have two parameters that define conditions on the returned nodes. The first parameter `user` specifies to return only nodes that have been accessed by the specified user. Analogously, we define the parameter `role`. Both parameters can be set to `any` to indicate to return nodes accessed by any user or in any role. Optionally, each parameter can be set to `current`. In this case, the current user or his current role is used for the check. For example, `created(any, current)` returns all nodes which have been created by users who were active in the same role as the one in which the current user is active in.

Getting Specific Nodes of Current Rule

We define three functions for accessing specific nodes within an XPath pattern. The function `ac:current-node` returns the node in question for which the XPath pattern is evaluated. This function is required when the pattern's context changes to a document that is different from the document for which the pattern was initiated. The function `ac:src-node` retrieves the source node in question when checking a copy rule. In a similar fashion, the function `ac:dest-node` returns the destination node of a copy rule. The last two functions are necessary to define copy rules which compare the source and destination objects with each other.

4.1.6 *System Architecture*

Applying our model in a scenario where multiple users concurrently edit multiple documents introduces four requirements. First, since access rights of one document depend on other documents, we need a method for accessing these distributed documents when calculating access rights. Second, changes to one document require the recalculation of the views of all dependent documents, which are currently viewed. The straight forward approach for this is to recalculate the views of all dependent documents after a document has been changed. However, this results in a much higher number of view recalculations compared to models which only define access depending on the currently edited document. For example, editing 20 depending documents concurrently, leads to a 20 times higher number of view recalculations with the straight forward approach. Therefore, we need a method which reduces the number of these view recalculations. Third, the changes by one user to a document can revoke the access rights of other users which are currently editing dependent documents. As a consequence, access rights can be revoked during an editing process, which can lead to conflicts regarding the content of the document and the access rights. Consequently, we need a method for handling these conflicts. Fourth, the aforementioned straight forward approach causes intermediate editing steps to become relevant for access decisions of other users, which is not desired. For example, a user can change a policy relevant element of a document by first deleting it and then replacing it with an updated version afterwards. In this example, the first step can revoke the access rights of another user, whereas the second step might restore these access rights.

In the following we present a system architecture that meets the four requirements mentioned above. Its components are explained in the following sections. Additionally, we describe the algorithms and protocols that are required for the interaction between the components.

4.1.6.1 *Overview*

Our system architecture and its components are depicted in Figure 4.4. Our system uses four databases. The document database (Doc DB) contains all the documents of the system. The rule database (Rule DB) contains the AC rules, which specify allowed or denied accesses to the documents and their parts. The copy database (Copy DB) stores the is-copy-of relation of the objects. Since the is-copy-of relation can be depicted by a graph, we speak of an edge when we refer to a single is-copy-of relation between two objects. Finally, the user database (User DB) stores the credentials of the users of the system as well as the corresponding roles including their hierarchy.

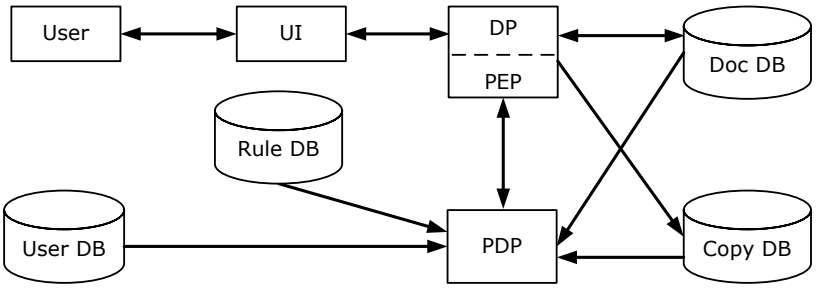

Figure 4.4: System Architecture for History-based Access Control

The user interface (UI) presents documents to the user and offers operations that can be performed on the documents. If the user invokes such an operation, the corresponding request is sent to the document processor (DP), which performs the requested operation if it is permitted. Inside the DP, the policy enforcement point (PEP) intercepts each operation and asks the policy decision point (PDP) whether the requested operation is allowed. The PDP uses the four databases to decide whether to allow or deny the requested operation. This architecture allows us to access distributed documents when a rule is evaluated and therefore it represents a solution for the first requirement mentioned in Section 4.1.6. In the following, we explain the workflow for editing a document to illustrate the processes within our architecture.

4.1.6.2 *Workflow*

A document must be opened before it can be viewed or edited. Therefore, the UI offers a command to open a document. This command is sent to the DP, which loads a copy of the document from the document database. We refer to this process as *check-out*, since it has semantics similar to the check-out command of a version control system [Tic85]. After the check-out, the user can edit the document by applying the operations of our model. The changed content of an opened document including the corresponding histories becomes relevant for access decisions of other documents after it is checked-in. Up to then, the content of the opened document is only relevant for access decisions concerning that document itself. The document and the corresponding histories are kept as a local copy in the DP. To *check-in* a document, the user must invoke the corresponding command of the UI. Then, the DP stores the copy of the document back to the document database.

The check-in and check-out concept is more efficient and offers a higher usability compared to directly working on the policy-relevant version of a document. The first concept is more efficient, because changed content must be propagated less often. In other words, we propagate the changes only when a document is checked-in and do not propagate them immediately after each change. This also reduces the overhead for recalculating permissions. The usability is also higher because of the transaction semantics of the approach. With this concept a user can decide when the changing of a document is done, instead of having potentially unwanted intermediate states become relevant for access decisions. With this concept we give a solution for the second and fourth requirement mentioned in Section 4.1.6.

In the following, we present and explain a set of algorithms for implementing the workflow sketched above.

4.1.6.3 *Check-out*

When a user invokes the command to check-out a document, the DP first loads a copy of that document from the Doc DB. The Doc DB maintains a list for each document that denotes by which users the corresponding document is currently opened to support concurrent access to documents. The PDP executes Algorithm 3 to create a view. This algorithm removes nodes from the document for which the user in question has no view permission and deleted nodes. For that purpose, the algorithm adds a marker to each node which is set initially to “default”, where a node can either be an element, an attribute or a text

block. Next, we sort all rules by their role and their mode. More special roles are priorized over less special roles and deny rules are placed before allow rules. Then, we remove inapplicable rules. For each of the remaining rules, the corresponding XPath pattern is evaluated. The result of this step is a set of nodes that match with the current XPath pattern, which defines the applicable objects of the rule. For each of these nodes, the marker is set according to the mode field of the current rule. If all nodes have a marker different from "default" we stop inspecting rules. Finally, we remove every node with a marker set to "default", every node with a marker set to "deny" and deleted nodes. After that, the PDP sends the view to the DP, which creates history entries for the view operation and forwards the view to the UI.

```
   Input  : rules_all, role_curr, role_hierarchy, doc
   Output: doc
 1 add marker to every node of doc
 2 set marker of every node of doc to "default"
 3 sort rules_all by role (special first) and mode (deny first)
 4 for each rule_i of rules_all do
 5     if operation of rule_i is not "view" or role of rule_i is not
       inferior or equal to role_curr then
 6         continue with next iteration of loop
 7     nodes_result ← evaluate XPath of rule_i for doc
 8     for each node_j of nodes_result do
 9         if marker of node_j is "default" then
10             set marker of node_j to mode of rule_i
11     if all markers of doc are different from "default" then
12         exit loop
13 for each node_j of doc do
14     if marker of node_j is "default" or "deny" or the node is
       deleted then
15         remove node_j and subtree below from doc
16 return doc
```

Algorithm 3: Create View

4.1.6.4 *Editing*

To edit a document, the user first selects an operation offered by the UI. This operation is sent to the DP, where the PEP intercepts the operation to check whether it is allowed. For this purpose, the PEP sends the requested operation together with the current document to the PDP, which evaluates the rules to answer the

request of the PEP. For this purpose, the PDP performs Algorithm 4.

```
   Input  : rules_all, role_curr, role_hierarchy, op, doc, obj,
            doc_dest, obj_dest
   Output: deny | allow
 1 sort rules_all by role (special first) and mode (deny first)
 2 for each rule_i of rules_all do
 3     if operation of rule_i is not op or role of rule_i is not inferior
       or equal to role_curr then
 4         continue with next iteration of loop
 5     if op is "copy" then
 6         nodes_result ← evaluate XPath for source of rule_i for
           doc
 7     else
 8         nodes_result ← evaluate XPath of rule_i for doc
 9     if obj is not contained in nodes_result then
10         continue with next iteration of loop
11     if op is "copy" then
12         nodes_result ← evaluate XPath for destination of rule_i
           for doc_dest
13         if obj_dest is not contained in nodes_result then
14             continue with next iteration of loop
15     return mode of rule_i
16 return "deny"
```

Algorithm 4: Evaluate Rules

The algorithm for rule evaluation sorts all rules like the previous algorithm. Then, it checks the applicability of each rule by inspecting its role and its operation. For each rule, the XPath pattern is evaluated to check whether it matches with the object in question. In the case of a copy operation (line 11), the XPath pattern for the destination is evaluated, too. If the rule is applicable, its mode is returned. After evaluating all rules, the algorithms returns "deny", if none of the rules was applicable. The PDP sends the result of this algorithm back to the DP. If the result is deny, the DP does not perform the requested operation and informs the user via the UI. If the result is allow, the DP performs the requested operation. For that purpose, it executes the algorithm for the selected operation. To give details on how the operations of our model are performed and how histories are recorded, we present the corresponding algorithms in the following.

Create

The create operation is used to create elements, attributes or text. In all cases, we add history information that describes the operation. Algorithm 5 is used for creating a new element. It creates the element itself and the corresponding history entry. Since the algorithm for creating a new attribute is very similar to the one for creating a new element, we refrain from describing it.

Input : doc_{dst}, elem_{dst}, subj, role, $\text{time}_{\text{curr}}$
Output:
1 $\text{elem}_{\text{new}} \leftarrow$ create new element at elem_{dst} in doc_{dst}
2 create element history for elem_{new}
3 `create_hist_entry(`*"Create elem."*, subj, role, $\text{time}_{\text{curr}}$`)`

Algorithm 5: Create Element

Algorithm 6 is used to add new text to an element. For this purpose, the DP first determines the creation context of the existing element and the current context. Then, both contexts are compared. If they are different the algorithm for splitting blocks is invoked. After that, a new text block is created to which the new content is added. Then, the corresponding history entry is created. If the contexts are equal, the new content is inserted into the element.

Input : doc_{dst}, elem_{dst}, subj, role, $\text{time}_{\text{curr}}$, content, $\text{pos}_{\text{insert}}$
Output:
1 `split_block(`doc_{dst}, elem_{dst}, $\text{pos}_{\text{insert}}$`)`
2 $\text{block}_{\text{dst}} \leftarrow$ `create_block(`doc_{dst}, elem_{dst}, $\text{pos}_{\text{insert}}$, content`)`
3 `create_hist_entry(`*"Create text"*, $\text{block}_{\text{dst}}$, subj, role, $\text{time}_{\text{curr}}$`)`

Algorithm 6: Create Text Content

Algorithm 7 splits text blocks. This is needed when text is created, copied or deleted. If position $\text{pos}_{\text{split}}$, at which the new text $\text{block}_{\text{old}}$ should be added, is not within an existing text block, the algorithm terminates immediately. In the other case, the existing text block $\text{block}_{\text{old}}$ has to be split at $\text{pos}_{\text{split}}$. Therefore, we remove the content after $\text{pos}_{\text{split}}$ from $\text{block}_{\text{old}}$. This removed content is added to the new text block $\text{block}_{\text{new}}$ which is positioned after $\text{block}_{\text{old}}$. Next, $\text{block}_{\text{new}}$ gets a copy of the history of $\text{block}_{\text{old}}$. If the split block was in an is-copy-of relationship with other blocks, the effect of the splitting has to be captured. For this purpose, the DP copies all edges of the Copy DB which either point to $\text{block}_{\text{old}}$

or originate from it and modifies them to point or originate from $block_{new}$. The DP stores these edges in its internal memory until the document is checked-in.

Input : doc_{dst}, $elem_{dst}$, pos_{split}
Output:
1 **if** pos_{split} *does not point within a block* **then**
2 **return**
3 $block_{old} \leftarrow$ text block where (doc_{dst}, $elem_{dst}$, pos_{split}) point into
4 content $\leftarrow$ text content of $block_{old}$ from pos_{split} to end
5 delete text content of $block_{old}$ from pos_{split} to end
6 $pos_{new} \leftarrow$ end of $block_{old}$
7 $block_{new} \leftarrow$ `create_block`(doc_{dst}, $elem_{dst}$, pos_{new}, content)
8 get subj, role, time from history entry of $block_{old}$
9 entries $\leftarrow$ copy of history entries for $block_{old}$
10 modify entries to reference $block_{new}$
11 $edges_{to} \leftarrow$ copy of edges of Copy DB that point to $block_{old}$
12 modify all edges in $edges_{to}$ to point to $block_{new}$
13 $edges_{from} \leftarrow$ copy of edges of Copy DB that originate from $block_{old}$
14 modify all edges in $edges_{from}$ to originate from $block_{new}$
15 add $edges_{to}$, $edges_{from}$ to global set of temp. edges

Algorithm 7: Split Text Block

Copy

The copy operation is used to copy elements or text. Algorithm 8 is used for copying elements.

Input : doc_{src}, $elem_{src}$, doc_{dst}, $elem_{dst}$, subj, role, $time_{curr}$
Output:
1 `create_element`(doc_{dst}, $elem_{dst}$, subj, role, $time_{curr}$)
2 $edge_{new} \leftarrow$ `create_edge`(doc_{src}, $elem_{src}$, doc_{dst}, $elem_{dst}$)
3 add $edge_{new}$ to global set of temp. edges
4 **for** *each* $attr_i$ *of* $elem_{src}$ **do**
5 **if** $attr_i$ *is not deleted* **then**
6 `create_attribute`(doc_{dst}, $elem_{dst}$, subj, role, $time_{curr}$, *name of* $attr_i$); `change_attribute`(doc_{dst}, $elem_{dst}$, subj, role, $time_{curr}$, *name of* $attr_i$, *value of* $attr_i$);

Algorithm 8: Copy Element

The algorithm first creates a new element at the destination document by applying Algorithm 5. After that, the DP captures this copy by adding a corresponding edge into its internal memory. The copied element gets all attributes including their values of its source element which have not been deleted.

Algorithm 9 is used to copy text. It splits both the text blocks at the source document and the text blocks at the destination. Then, it creates a new block at the destination document for each of the copied blocks. It creates and stores the corresponding edge reflecting the copy. The new block gets its first history entry describing its creation context.

Input : doc_{src}, elem_{src}, $\text{text}_{\text{start}}$, text_{end}, doc_{dst}, elem_{dst}, $\text{pos}_{\text{insert}}$, subj, role, $\text{time}_{\text{curr}}$
Output:

```
1  split_block(doc_src, elem_src, text_start)
2  split_block(doc_src, elem_src, text_end)
3  split_block(doc_dst, elem_dst, pos_insert)
4  for each block_src within text_start and text_end of elem_src do
5      content ← text content of block_src
6      block_dst ← create_block(doc_dst, elem_dst, pos_insert,
       content)
7      edge_new ← create_edge(doc_src, elem_src, block_src,
       doc_dst, elem_dst, block_dst)
8      add edge_new to global set of temp. edges
9      create_hist_entry("Create text", doc_dst, elem_dst,
       block_dst, subj, role, time_curr)
10     pos_insert ← end of block_dst
```

Algorithm 9: Copy Text Content

Change Attribute and Delete

Changing an attribute value is also logged with an entry in the history of the element containing that attribute. The corresponding algorithm is similar to the algorithm for creating an element. The delete operation can be applied to elements, attributes and text. Since the required algorithms are similar, we explain only Algorithm 10 for deleting text. If start or end of the text to be deleted point into a block, the affected blocks have to be split to keep track of the deleted parts. Each deleted text block gets a final history entry to indicate its deletion. This history entry is used to exclude the deleted block from future views.

Since performing an operation can lead to modifications of view permissions, the DP asks the PDP to update the view as

Input : doc_{src}, $elem_{src}$, $text_{start}$, $text_{end}$, doc_{dst}, $elem_{dst}$, subj, role, $time_{curr}$
Output:
1 `split_block`(doc_{src}, $elem_{src}$, $text_{start}$)
2 `split_block`(doc_{src}, $elem_{src}$, $text_{end}$)
3 **for** *each* $block_{src}$ *within* $text_{start}$ *and* $text_{end}$ *of* $elem_{src}$ **do**
4 `create_hist_entry`("*Delete text*", doc_{dst}, $elem_{dst}$, $block_{dst}$, subj, role, $time_{curr}$)

Algorithm 10: Delete Text Content

described above. The updated view is generated by executing Algorithm 3. The result is presented to the user via the UI.

4.1.6.5 *Check-in*

A user can invoke the check-in command of the UI to save his changes to an opened document doc_a, which is currently stored only within the DP, to the Doc DB. As a result of this, the checked-in version of the document becomes relevant for the access decisions of other documents, which also includes concurrently opened versions of doc_a. For these documents the permissions must be recalculated, which possibly revokes permissions of currently edited documents. The concurrent editing of a document can also lead to conflicts, where the editing of one user to doc_a is incompatible to the editing of another user, who also has edited doc_a. For these reasons, we have to perform two steps when a document is checked-in. In step one, we have to solve conflicts between the concurrent versions of a document. In step two, we must update the permissions of other affected documents whose permissions depend on the saved document.

To perform step one, we first retrieve the list of concurrently edited versions of doc_a, which is maintained by the Doc DB for each opened document. Next, we must merge all concurrently edited versions of doc_a to one consistent version. We apply a conflict resolution strategy to solve conflicts between concurrently edited documents. It depends on the scenario to define a specific strategy. One possible strategy is to resolve conflicts manually. An automatic strategy can accept or reject changes depending on the role of the subject that performed the changes or depending on the time the changes were performed, since this information is available in the corresponding histories. After the conflicts are solved, the temporarily stored edges, which correspond to the accepted operations, are saved to the Copy DB.

To perform step two, we first inspect the Copy DB to retrieve the opened documents that might depend on doc_a. These docu-

ments have at least one node, that is in is-copy-of relation with a node of doc_a. Then, we recalculate the permissions of these documents for their current users. In some cases, permissions of edited nodes are revoked. In these cases, the UI asks the user whether he wants to reject the current changes or keep them and accept being unable to make further changes. These two steps provide a solution for the third requirement mentioned in the introduction.

4.1.7 *Modeling Access to Standards*

In this section, we model our scenario described in Section 4.1 using the history-based approach that we have presented in the previous sections. We assume that the role hierarchy depicted in Figure 4.5 is used and that all subjects act in one of these roles. We use the generic role employee to describe an employee of any of the three companies of our example. We use special roles for the employees of each company and derive these roles from the generic role employee. For example, we use the role employee-smartchips as a role for employees of the company SmartChips. Moreover, we assume that every XML document

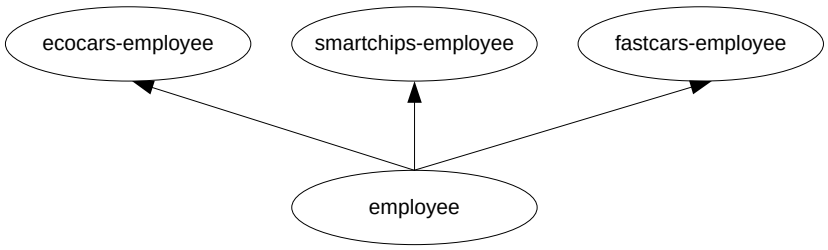

Figure 4.5: Role Hierarchy

defining a standard has a `class` attribute at its root element. This attribute defines which company the standard belongs to or that the standard is public. In the first case, the attribute is set to the name of the corresponding company, whereas in the second case, the attribute is set to public. We assume that a security administrator maintains the attribute `class` and that other subjects are not allowed to change this attribute.

For each statement of the scenario, which defines an allowed or forbidden action, we need to specify one or more AC rules. Because of the default semantics of our model, all unspecified operations are denied. As a consequence, we only need to define allow rules and, in addition, exceptions from these allow rules.

First of all, we need a set of rules to define the fact that employees of a company are allowed to view and edit the standard

of that company. Rule 4.6 is an example of such a rule. This rule states that employees of the company SmartChips are allowed to view documents that have a root element with the attribute `class` set to SmartChips. In other words, employees of SmartChips are allowed to view their own standard. Similar rules must be defined for the remaining operations (`create`, `delete`, `copy` and `change-attribute`) and for the other two companies (EcoCars and FastCars).

Role	`employee-smartchips`
Operation	`view`
Object	`/*/@class == 'SmartChips'//*`
Mode	`allow`

Figure 4.6: Allow Employees of SmartChips to View their own Standard

Next, we define a rule that states that every employee is allowed to view parts that were copied from a public standard. In Rule 4.7, we use the functions `ac:copies` to determine where the part was originally copied from, where the namespace of our functions is indicated by the prefix `'ac:'`. This function returns all nodes that are in is-copy-of relation with the current node. The returned list of nodes is sorted according to the time of creation of the nodes, in ascending order. We can retrieve the first node of this list with the expression `ac:copies()[1]`. This is the original node from which all other nodes on the list have been copied. We inspect the attribute `class` of the corresponding document containing the first node and check whether the attribute is set to public. In this case, the viewing is allowed.

Role	`employee`
Operation	`view`
Object	`fn:root(ac:copies(.)[1])/` `@class == 'public'`
Mode	`allow`

Figure 4.7: Allow Viewing if the Object is Copied from a Public Standard

In addition to the previous rule, we define Rule 4.8, which allows the copying of parts of public standards. In this rule, we also use the function `ac:copies`. Using this function offers the advantage that public parts within a company's standard can also be copied. For example, this rule allows the copying of parts

of a public standard that reside in the standard of SmartChips. Moreover, this rule defines no restriction for the destination. In other words, it allows copying parts of a public standard to anywhere. Since our model requires a view permission on an object to perform another operation, a subject can only copy parts to locations for which it also has the view permission. This restriction avoids the possibility that EcoCars could copy parts of a public standard to FastCars' standard, since employees of EcoCars have no permission to view the standard of FastCars.

Role	`employee`
Operation	`copy`
Object	`fn:root(ac:copies(.)[1])/ @class == 'public'`
Destination	`//*`
Mode	`allow`

Figure 4.8: Allow Copy if Source is Public

We stated in Section 4.1 that employees of SmartChips are allowed to view both, the standard of EcoCars and the standard of FastCars. Rule 4.9 expresses this statement by inspecting the attribute class of the corresponding root element of the element in question.

Role	`employee-smartchips`
Operation	`view`
Object	`//*[(/*/@class == 'EcoCars') or (/*/@class == 'FastCars')]`
Mode	`allow`

Figure 4.9: Allow SmartChips' Employees to View other Standards

Recall that we did not restrict the destination of copy operations in Rule 4.8. In addition, Rule 4.9 adds view permissions on FastCars' standard for employees of SmartChips. These added view permissions enable employees of SmartChips to copy parts of public standards to the standard of FastCars, since view permissions are required to perform any other operation. However, employees of SmartChips should not be allowed to modify FastCars' standard. As a consequence, we need a rule that prevents unwanted copy operations. For this purpose, Rule 4.10 explicitly denies all transfers to FastCars' standard for employees of

SmartChips. Rule 4.10 takes precedence over Rule 4.8, since deny rules take precedence over allow rules.

Role	`employee-smartchips`
Operation	`opy`
Object	`//*`
Destination	`//*[/*/@class == 'FastCars']`
Mode	`deny`

Figure 4.10: Deny Copying to FastCars' Standard by SmartChips

As stated in the scenario, SmartChips' employees are allowed to copy parts of their own standard to the standard of EcoCars. Rule 4.11 fulfills this condition. It inspects the original location of the parts instead of their current location. We retrieve the standard where the part was created by using the `ac:copies` function. With this rule, we also allow the employees of SmartChips to copy parts of their standard that are located in the standard of EcoCars to another location within EcoCars' standard.

Role	`employee-smartchips`
Operation	`copy`
Object	`fn:root(ac:copies(.)[1])/ @class == 'SmartChips'`
Destination	`//*[/*/@class == 'EcoCars']`
Mode	`allow`

Figure 4.11: Allow SmartChips' Employees to Copy to EcoCars Standard

As described in the scenario, employees of SmartChips are allowed to copy parts from FastCars' standard to their own standard. Rule 4.12 implements this condition and uses similar mechanisms to Rule 4.11. In this case, we also inspect the original location of these parts, which also allows copying them from other locations, e.g., from SmartChips' standard.

Under their agreement with EcoCars, employees of SmartChips need to edit the standard of EcoCars. As a consequence, we must define the corresponding permissions by a set of rules. We need a rule for create, delete and change attribute. Rule 4.13 is the corresponding rule for the create operation. Since the other two rules differ only in their operation field, we refrain from listing these rules.

Role	`employee-smartchips`
Operation	`copy`
Object	`fn:root(ac:copies(.)[1])/ @class == 'FastCars'`
Destination	`//*[/*/@class == 'SmartChips']`
Mode	`allow`

Figure 4.12: Allow SmartChips' Employees to Copy from FastCars

Role	`employee-smartchips`
Operation	`create`
Object	`//*[/*/@class == 'EcoCars']`
Mode	`allow`

Figure 4.13: Allow SmartChips' Employees to Edit the Standard of EcoCars

When editing EcoCars' standard, employees of SmartChips should not delete objects or modify attributes that have not been created by them. Rule 4.14 implements this restriction by checking the creation context of an object before granting a delete operation. The corresponding context is retrieved with the function `ac:getCreationContext()`. Then, we inspect the role attribute of the retrieved context. An employee is not allowed to delete or change an object if he did not create this object. In addition, we need a similar rule for denying the modification of attributes by employees of SmartChips. Since this rule differs only in its operation field from Rule 4.14, we refrain from presenting it.

Role	`employee-smartchips`
Operation	`delete, change-attribute`
Object	`//*[ac:getCreationContext()/ @role != employee-smartchips]`
Mode	`deny`

Figure 4.14: Deny Deleting Elements and Changing Attributes by SmartChips

Since employees of SmartChips can both read the standard of FastCars and edit the standard of EcoCars, they could transfer knowledge from EcoCars to FastCars. There are several possibili-

ties for defining restrictions to prevent this. We can model this condition in three alternative ways, for which we present AC rules in the following paragraphs:

1. After an employee of SmartChips has accessed the standard of Fast-Cars, we deny him the viewing of EcoCars' standard. Since view permissions are required for any other operation, employees of SmartChips cannot perform any operation on EcoCars' standard in this case.

2. After an employee of SmartChips has accessed the standard of Fast-Cars, he is still allowed to view the standard of EcoCars. However, editing of the standard is denied.

3. After an employee of SmartChips has accessed the standard of Fast-Cars, he is still allowed to edit the standard of EcoCars. However, copying of parts of FastCars' standard to EcoCars' standard is denied.

Within each of these three alternatives, we have two options regarding which subjects we define the restrictions for:

1. We can restrict access for all employees of SmartChips after any of them has accessed the standard of FastCars.

2. We can restrict access only for the specific subject that has accessed the standard of FastCars.

We start by modeling the first alternative. As stated above, we have two options for how to model this alternative. The most restrictive option is to deny any access to the standard of EcoCars after any employee of Smart-Chips has accessed the standard of FastCars. Rule 4.15 implements this version of the scenario.

Role	`employee-smartchips`
Operation	`view`
Object	`//*[(/*/@class == 'EcoCars') and (fn:count(ac:accessed(any, current)[/*/@company == 'FastCars']) > 0)]`
Mode	`deny`

Figure 4.15: Deny Access to EcoCars' Standard for SmartChips

This rule revokes the view permissions, which are needed as a basis for any other operation. The rule checks two conditions. First, the corresponding object must reside in the standard of EcoCars. The second condition uses our `ac:accessed` function

to check whether any employee of SmartChips has previously accessed FastCars' standard. This function has the parameters `user` and `role` that define conditions on the returned nodes. The first parameter `user` specifies that only nodes that have been accessed by the specified user should be returned. Analogously, we define the parameter `role`. Both parameters can be set to `any` to indicate that nodes accessed by any user or in any role should be returned. In Rule 4.15, we use `any`, `current` as parameters for the `ac:accessed` function, which indicates that all objects accessed by any subject who was active in the same role as the current subject should be returned. As a result, all objects that were accessed by an employee of SmartChips are returned. If we use `current`, `any` as parameters for the `ac:accessed` function, we get different semantics. In this case, all objects that were accessed by the current subject are returned. As a result, access is blocked only for an individual employee, while the access of other employees is not blocked. The requirements of the scenario determine which of the two options is chosen. If access is restricted at the level of employees, we retain more flexibility since other employees of SmartChips can continue to work on EcoCars' standard while only employees that have accessed FastCars' standard are not permitted to edit it. On the other hand, if the information in the FastCars' standard is very confidential, this flexible version might not be sufficiently restrictive; employees of SmartChips could pass on their knowledge outside the IT system, and a different employee could transfer the confidential information. In such cases, the more restrictive version should be applied.

Next, we explain how to express the second alternative with our AC rules. Instead of denying any access, we can deny edit access only. To do this, we need to define rules similar to Rule 4.15, where we need one rule for each of the editing operations `create`, `delete`, `change-attribute` and `copy`. In a similar fashion to the case where we denied the viewing, we can deny the editing both at the level of employees as well as at the level of the entire company. Rule 4.16 and Rule 4.17 deny editing access for all employees of SmartChips if any employee of Smart-Chips has accessed FastCars' standard. Alternatively, editing can be denied for only the specific employee if the parameters of the `ac:accessed` function are set to `current`, `any`.

The third alternative is the least restrictive one. In this case, we only deny explicit transfers of parts of FastCars' standard to EcoCars' standard. Rule 4.18 is the AC rule for this alternative. It is the least restrictive version of the three sketched alternatives since both viewing and editing are still allowed.

Role	`employee-smartchips`
Operation	`create, delete, change-attribute`
Object	`//*[(/*/@class == 'EcoCars') and (fn:count(ac:accessed(any, current) [/*/@company == 'FastCars']) > 0)]`
Mode	`deny`

Figure 4.16: Deny Editing by SmartChips after Access to FastCars' Standard

Role	`employee-smartchips`
Operation	`copy`
Object	`//*[fn:count(ac:accessed(any, current) [/*/@company == 'FastCars']) > 0]`
Destination	`//*[/*/@class == 'EcoCars']`
Mode	`deny`

Figure 4.17: Deny Copying by SmartChips after Access to FastCars' Standard

The three alternatives show that our history-based approach is sufficiently expressive to offer different ways for defining access to standards. In [RTE07b], we have explained how our model can be used to enforce the policies of the Chinese Wall model proposed by Brewer and and Nash [BN89]. The resulting rules provide the same level of security as the original Chinese Wall model while being less restrictive. Thus, we believe that our model provides a flexible and expressive method to define access depending on histories.

In this section, we focused on defining rules for controlling access to XML documents. In the next section, we consider scenarios, in which customers are involved in the execution of a special kind of business processes, such as selling and consuming digital goods. In these cases, digital goods have to be protected from being used in an unauthorized way, i.e., being shared in public networks. This requirement reflects another facet of usage control. In this context, we have to define usage rules for digital content, e.g., transferring music files, and we show how to enforce such kinds of rules.

Role	`employee-smartchips`
Operation	`copy`
Object	`//*[fn:root(ac:copies(.)[1])/ @class == 'FastCars']`
Destination	`//*[/*/@class == 'EcoCars']`
Mode	`deny`

Figure 4.18: Deny Copying after Access to FastCars' Standard

4.2 CONTROLLED SHARING OF DIGITAL CONTENT

Efficient compression algorithms and high bandwidths allow companies to set up business models based on selling digital content, such as digital music and electronic books. However, the same technical possibilities allow users to share digital content over communication networks at low cost. This led to the emergence of file sharing networks over which users illegally distribute digital content. The content owner and the content provider respond to that illegal distribution with the concept of Digital Rights Management (DRM) systems. Basically, DRM aims at controlling the usage of digital content. DRM is sophisticated and cannot been analyzed in isolation [Rum03], therefore there are different definitions of it [Ian01, LSNS03]. [RMT01] proposes a popular reference model for a DRM system, which is depicted in Figure 4.19. The following outlines the process flow depicted in the DRM reference model:

1. User obtains content.
2. User attempts to use, i.e., to render, the content in some way. This triggers the DRM controller. Once activated, the DRM controller gathers information necessary for generating a license. This includes identity information for the user and/or client device and information from the content package, including the content identifier.
3. DRM-client makes rights request.
4. The license server verifies the submitted client identification or attributes against an identity database.
5. The license server looks up rights specifications (usage rules) for the content.
6. The license generator pulls together rights information, client identity information, and encryption keys, and cre-

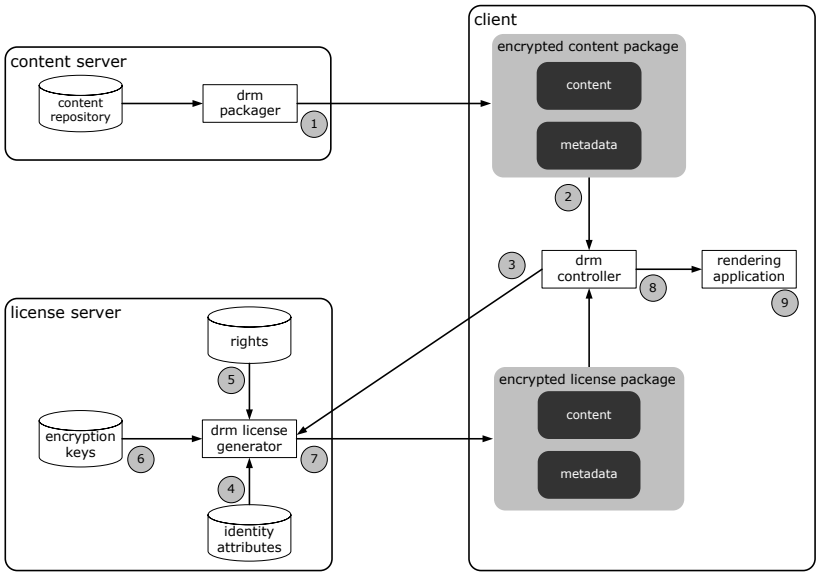

Figure 4.19: DRM Reference Model [RMT01]

ates a license, which is itself encrypted or at least tamper proofed.

7. The license is sent back to the client.

8. After the license has been generated and any authentication steps are complete, the DRM controller can decrypt the content and release it to the rendering application.

9. Finally, the rendering application plays or shows the content to the user.

The process sketched above illustrates the basic idea behind DRM systems. They enable content providers to define usage rules with the help of Rights Expression Languages (RELs). RELs consist of a grammar (a schema), also called the language concept [Gut03], that describes the basic grammar for the expression of rights terms and conditions, and their relation to the assets in question and the parties involved. The resulting right definition is referred to as a license object. Since this object defines the usage of a certain content for a certain customer, the rights object itself and the application processing, i.e., DRM controller and rendering application, should be protected against manipulation. Otherwise, the content could be used in a way other than defined by the content provider.

With respect to their purpose, RELs are similar to our Access Control rules presented in Section 4.1.4. Both define access to digital objects. However, since they are meant to support different

usage scenarios, they differ significantly regarding the addressing of objects and subjects.

Existing DRM systems have two shortcomings. First, they build their protection measures on the basis of unreliable software-based solutions, such as obscurity techniques. Publicly available documentation for common rights management systems from Microsoft [Mico8] or Apple [Appo8] do not disclose how they resist replay attacks for their (proprietary) dynamic license implementations. Obscurity techniques are as long effective as long as the obscurity is undiscovered. Moreover, most of the current DRM solutions are closed software and cannot be verified for inherent security flaws. In addition, this approach may lead to unwanted side-effects. A prominent example for these unwanted side-effects is the Sony BMG CD copy prevention scandal [Hano6]. Sony BMG included a software for copy prevention on music CDs. This software was automatically installed on Windows computers when customers tried to play their CDs. This installation has proved to be particularly problematic, since the installed software interferes with the normal way in which the Windows operating system plays CDs and creates security holes that can be exploited by malicious software such as worms or viruses [Ruso5].

Second, existing DRM systems have been developed from the content providers' point of view. Consequently, they do not regard consumers' expectations. As stated in [Feto3], the neglect of consumers' expectations may hamper the overall acceptance of DRM. According to the survey [DSVWo5], 75% of the consumers want to share music with their friends and families and would pay more for this kind of usage.

Against this background, we realize that a successful DRM system has to respect consumers' expectations while providing reliable security mechanisms to protect digital content from unauthorized usage.

Trusted computing technologies provide the necessary technical foundation for building reliable systems which behave in an expected way. They basically provide a set of techniques to enforce security policies. Although trusted computing has a bad reputation, we believe that this technology has the necessary potential for a wide range of sensitive applications. We agree with [KGo3], which states that trusted computing technologies are not the same as DRM systems. In the following, we first give an introduction to trusted computing technologies. Then we propose a protocol which can be used for developing a robust DRM system that supports transfer of licenses to other users or devices. We the present such a DRM system in Section 4.2.7.

4.2.1 *Background on Trusted Computing Mechanisms*

The increasing complexity of IT-systems is a major factor for the growing number of system vulnerabilities. Trusted Computing techniques [BCP+03] offer one possibility to overcome this shortcoming by providing a tamper-resistant hardware component that provides a root of trust. The introduction of the Trusted Platform Module (TPM) which is specified by the Trusted Computing Group (TCG) provides vital functions for IT-security. Besides the potential of offering a protected storage for storing cryptographic keys, the TPM also offers the possibility to attest the configuration of the local platform to a remote platform which is called remote attestation (RA). This process is of particular interest in the context of DRM [LSNS03] or Enterprise Security [SvDW04] to ensure that the client platform is trustworthy and is behaving in accordance with a defined policy. This becomes relevant when sensitive data is transferred which should only be used in an authorized way, i.e. the content can be used by an authorized user on a specified platform. For this purpose, a policy can be defined which is enforced by the local client software. If the policy enforcement mechanism is executed on the client-side with the help of software, an attacker can deactivate this mechanism by modifying the local client software.

To detect these kinds of attacks one may attest the status of the client platform using the RA before sending sensitive data to the client. Thus, the sender has a confirmation about the trustworthiness of the platform in question. However, RA protocols are not per se resident against masquerading attacks. These attacks are in [GPS06] referred to as relay attacks. An attacker who is in control of one malicious and one honest client may bypass the remote attestation by spoofing his malicious client to be the honest one. The attacker simply forwards all attestation queries and sends them to one client, which is in a trustworthy system state. The honest client sends the answer back, which must only be transferred back to the requester. This attack is a masquerading [RMAW99] attack, with the extension that the attacker is also in possession of the honest client. But, as we will see in the following sections, this is not an essential requirement. After this attack the malicious client platform state has then been approved trustworthy. Afterwards, the protected digital content is transferred to the pretended trusted platform, that does not enforce any policies to protect the content.

We believe that the masquerading attack is critical in the context of DRM systems. In this case, the content provider must be sure, that the policy, which defines the usage permissions, is

enforced correctly on the DRM-client. Therefore, the content provider first attests the state of the platform before he delivers protected digital content. This content is encrypted to bind it to the attested machine. However, the content provider is never sure that the key which is used for binding the content is stored on the attested client and is not on a second machine. In the latter case, the policies for protecting the content from being used in an unauthorized way will not be enforced.

The core of the TCG mechanisms [Tru08, BCP+03] is the Trusted Platform Module (TPM), which is basically a smartcard soldered on the mainboard of a computer. The TPM serves as the root of trust, because its hardware implementation makes it difficult to tamper with, and therefore it is assumed to be trustworthy. One must also assume that the hardware vendor is trustworthy and has designed the TPM chip according to the specification. Although the TPM chip is not specified to be tamper-resistant, it is tamper-evident, meaning that unauthorized manipulations can be detected.

The TPM can create and store cryptographic keys, both symmetric and asymmetric. These keys can either be marked migratable or non-migratable, which is specified when the key is generated. In contrast to non-migratable keys, migratable keys can be transferred to other TPMs. Due to its limited storage capacity, the TPM can also store keys on the hard disk. In this case, these keys are encrypted with a non-migratable key, assuring the same level of security as if the keys were stored directly in the TPM. The TPM is able to perform calculations on its own, e.g., it can use the generated keys for encryption and decryption.

In the context of this thesis, the Platform Configuration Registers (PCRs) are of particular interest. These registers are initialized on power up and are used to store the software integrity values. Software components are measured by the TPM and the corresponding hash-value is then written to this platform configuration register by extending the previous value of a specific PCR. The following cryptographic function is used to calculate the values for the specific registers:

$$Extend(PCR_N, value) = SHA1(PCR_N || value)$$

For every measured component an event is created and stored in the stored measurement log (SML). The PCR values can then be used together with the SML to attest the platform's state to a remote party. To make sure that these values are authentic, they are signed with a non-migratable TPM signing key, the Attestation Identity Key (AIK). The remote platform can compare

these values with reference values to see whether the platform is in a trustworthy state or not.

The TCG assumes a trusted operating system which is not part of the TCG specifications. The assumed trusted operating system measures the hash value of every process started after the boot process. For a description of a trusted operating systems confer [GPC+03, SZJD04].

The TPM additionally offers a number of different signing keys. One major key is the Endorsement Key (EK) which is generated by the module manufacturer and injected into the TPM. The EK uniquely identifies the TPM and is used to prove that the TPM is genuine. In addition, the EK is used to obtain an Attestation Identity Key (AIK). An AIK is created inside the TPM, signed with the private portion of the EK, and the public part is transferred to a third party (a Privacy-CA). The Privacy-CA verifies that the platform is a genuine TPM and creates a certificate which binds the identity key to the identity label and generic information about the platform. This certificate, also known as identity credential is sent to the TPM and later used to attest the authenticity of a platform configuration.

To reduce the amount of non-volatile memory required inside the TPM, it acts as an access control device for externally stored keys rather than storing all keys itself. For this purpose, the TPM generates the storage root key (SRK). This is a 2048-bit RSA key and is created when an user takes ownership of the TPM. The private part of SRK remains permanently in the TPM. The SRK is used to encrypt all other externally stored keys. This strategy offers the same security level as if all keys were stored internally.

4.2.2 *Remote Attestation*

The remote attestation is used to attest the configuration of an entity to a remote entity. This procedure is widely used to get integrity information before a client proceeds with the communication in order to use a service or receive data, e.g., digital content. This mechanism is referred as integrity reporting and can be applied in many scenarios and different applications, such as controlling access to a network depending on the trustworthiness of the client [SJZvD04]. The integrity reporting mechanism is also a required mechanism in the context of DRM applications, since it is obviously required that the DRM-client software is in a trustworthy state and executes a certain policy to prohibit unauthorized use, copy or redistribution of intellectual property [RC05].

[SZJvD04] states that the goal of integrity reporting protocols is "to enable a remote system, i.e., the challenger, to prove that a program on another system, i.e., the attesting system owned by the attestor, is of sufficient integrity to use". In this section, we discuss an integrity reporting protocol proposed by [SZJD04], which is based on the challenge-response authentication [BM92] and is used to validate the integrity of an attesting system.

Figure 4.20 illustrates the remote attestation of B against A and provides the background information on integrity reporting protocols. In step 1 and 2, A creates a non-predictable nonce and sends it to attestor B. In step 3a, the attestor loads the Attestation Identity Key from the protected storage of the TPM by using the storage root key (SRK). In the next step, the attestor performs a TPM_Quote command, which is used to sign the selected PCRs and the provided nonce with the private key AIK_{priv}. Additionally, the attestor retrieves the stored measurement log (SML). In step 4, the attestor sends the response consisting of the signed Quote, signed nonce and the SML to A. The attestor also delivers the AIK credential which consists of the AIK_{pub} that was signed by a Privacy-CA.

1.	A	: create 160 bit nonce
2.	A $\rightarrow$ B	: ChallengeRequest(nonce)
3a.	B	: load AIK_{priv}
3b.	B	: retrieve Quote $= Sig(AIK_{priv}, (PCR, nonce))$
3c.	B	: get SML
4.	B $\rightarrow$ A	: ChallengeResponse(Quote, SML) and $cert(AIK_{pub})$
5a.	A	: validate $cert(AIK_{pub})$
5b.	A	: validate Quote
5c.	A	: validate nonce and SML using PCR

Figure 4.20: Integrity Reporting Protocol [SZJD04]

In step 5a, A validates if the AIK credential was signed by a trusted Privacy-CA thus belonging to a genuine TPM. A also verifies whether AIK_{pub} provided by B is still valid by checking the certificate revocation list of the trusted issuing party. This step was also designed to discover masquerading by comparing the unique identification of B with the system identification given in AIK_{pub}. Nevertheless, this verification does not discover masquerading attacks as we will see in the next section.

In the next step, A verifies the signature of the Quote and checks the freshness of Quote in step 5c. Based on the received

SML and the PCR values A processes the SML and recomputes the received PCR values. If the computed values match the signed aggregate, the SML is valid and untampered. A now only verifies if the delivered integrity reporting values match the given reference values, thus A can decide if the remote party is in a trustworthy system state.

According to [SZJD04], the protocol described above should be resistant against replay, tampering and masquerading attacks. However, an attacker can successfully perform a masquerading attack on that protocol. We describe this attack in the following section.

4.2.3 *Masquerading Attack Scheme*

The attacker considered here has two platforms under his control. One platform runs a trustworthy operating system with the client software that enforces a certain policy, e.g., the DRM-client. This platform (C) represents the client that is conform to the original client software and therefore untampered. This platform is also equipped with a genuine TPM that supports the policy enforcement of the DRM-client. The attacker is also in control of one malicious client platform (M) that wants to gain control of protected digital content. We refer to this client as malicious client, since his enforcement mechanism has been tampered with and it is not conform to the original client software. We require for our attack that C answers the request from M, i.e. C is not configured to answer only requests from A. This is a necessary requirement for the success of the attack and is discussed in detail in section 4.2.4.

In our attack scheme, the attacker bypasses the remote attestation of M, by using the platform configuration of the honest client C to attest his malicious client running on M.

Figure 4.21 depicts the attack against the integrity reporting protocol. The challenging party A wants to securely validate the integrity of the attesting malicious system M. The malicious system M itself transfers all messages from A to the honest client C. The protocol is the same as the protocol shown in Figure 4.20 except for step 2a and 4, in which only the messages are forwarded.

Figure 4.22 simplifies the attack on the presented protocol by reducing it to the transferred messages. This figure shows the challenger (platform A) that wants to attest the client (M) before protected data is transferred. Platform M and C are working collaboratively. Platform M is authenticated on the basis of the provided information through platform C, this is simplified

1.	A	: create 160 bit nonce
2.	A $\rightarrow$ M	: ChallengeRequest(nonce)
2a.	M $\rightarrow$ C	: ChallengeRequest(nonce)
3a.	C	: load AIK_{priv}
3b.	C	: retrieve Quote $= Sig(AIK_{priv}, (PCR, nonce))$
3c.	C	: get SML
4.	C $\rightarrow$ M	: ChallengeResponse(Quote, SML) and $cert(AIK_{pub})$
4a.	M $\rightarrow$ A	: ChallengeResponse(Quote, SML) and $cert(AIK_{pub})$
5a.	A	: validate $cert(AIK_{pub})$
5b.	A	: validate Quote
5c.	A	: validate nonce and SML using PCR

Figure 4.21: Attacking the Integrity Reporting Protocol [STRE06]

through step 5 and 6. The figure also shows that the attack cannot be prevented with the use of a secure mutual authenticated TLS channel, since this protocol only authenticates subjects, i.e., a client or a server. The protocol does not consider AIKs. An attacker may be able to extract the X.509 certificate and the corresponding keys for a mutual authentication. Since standard TLS-applications, e.g., web browsers, support the export of the keys by the platform owner, he may transfer the certificates and the private keys from C to the non-conformant host M. The server (platform A) then authenticates the client based on the certificates and sends an attestation request to the malicious platform, which answers the request in the previously described manner. The shortcoming of the integrity reporting protocol is caused by the restricted usage possibilities of AIKs. According to [Tru08], the AIKs can exclusively be used for proving the authenticity of a platform. This means AIKs can neither be directly used to establish secure channels nor to authenticate communication partners. Therefore, the challenger cannot be sure whether the received message belongs to the attesting system. He only knows that he received a message from a genuine TPM.

Additionally, the possession of multiple AIKs is possible, the only requirement is, that the corresponding certificate must be certified by a trusted Privacy-CA thus belonging to a valid EK. As a consequence, the challenging party cannot verify the integrity of a platform only with the help of the AIKs belonging to that platform.

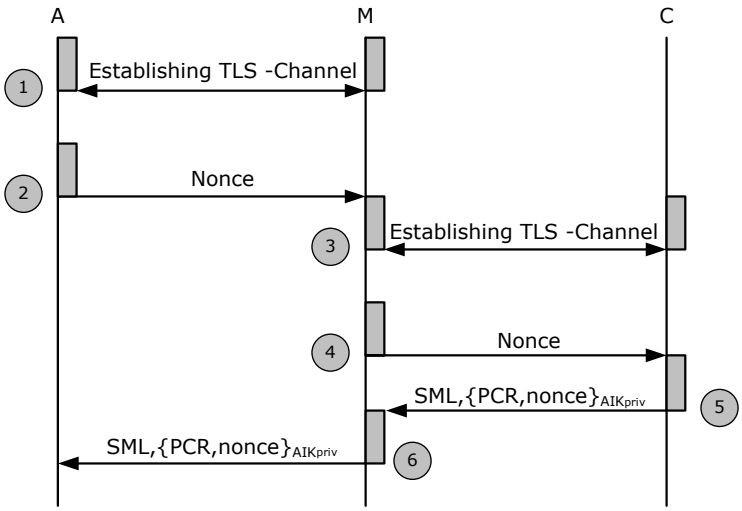

Figure 4.22: Collaborative Masquerading Attack on RA

Even if the server has the AIK_{pub} certificate and forbids the creation of new AIKs, masquerading attacks cannot be prevented, since the attesting system M pretends that it is in possession of the corresponding AIK_{priv} by using the integrity values with a valid AIK signature from platform C (Steps 5 and 6 in Figure 4.22). The challenging system is therefore not able to detect this attack in step 5a (Figure 4.21), since the attesting system delivers all information that identifies it as platform C. After the platform is authorized, the malicious platform is assumed to be trusted.

We explain the relevance of the masquerading attack by a concrete usage scenario, i.e. the initialization phase of a DRM-client. This phase includes steps 2, 3 and 4 shown in Figure 4.19. At this time, the DRM-client has not yet exchanged a cryptographic key with the server. This key is needed to bind the digital content to the client platform by means of encryption. Before exchanging the key, the server attests the client platform state to ensure that the client is running a trustworthy DRM-client software. If the attestation is successful the DRM-client software supports the user by the creation of an account and an appropriate cryptographic key on the client machine. This key is later used to establish a mutual authenticated channel (e.g. TLS-channel) which also authenticates the user. After the initialization, the server encrypts the digital content using the client key and transfers it to the client. However, this key was generated on a non-conformant platform, on which the enforcement of usage rules are bypassed.

4.2.4 *Evaluation of Possible Approaches*

In this section, we present some possible approaches to overcome the previously described shortcoming of integrity reporting protocols. However, we explain for each of these approaches why they fail to prevent the described attack.

Verifying Attestation Challenges

Since the attacker cannot modify the software on the honest platform, the honest platform may decide which attestation challenge (represented by a nonce) it will answer and which it will reject. For this purpose, the challenger could sign the attestation challenge and the prover validates the signature using a preinstalled certificate before he answers the challenge. In this case, the server's certificate must be included in the integrity measurement of the client software to detect a manipulation on that certificate.

However, any arbitrary software component that is able to answer attestation challenges can respond to the malicious challenge. An attacker can use such a software to circumvent the verification of the challenge of the client software. However, the verifying party detects that this software component is listed in the SML, where the correct software is also listed. On the downside, the SML does not tell which software issued the answer to the attestation challenge. The attacker can either use a software supplied by himself to answer the attestation challenge or he can misuse an existing application that serves a legitimate purpose. In the latter case, the issuing software appears to be trusted, because its software vendor might offer valid SML-reference values.

Regardless of which client software responds to the attestation challenge, the only requirement is that the nonce, which was sent from the server-application (platform A) is used for the attestation. Hence, bugs or design flaws that cause any arbitrary attestation challenge to be answered, affect other trusted client software. One approach is to forbid the installation of other client software components. However, this leads to high restrictions which can hardly be accepted in open environments.

Using Shared Secrets

One method may be to exchange a shared secret key between server and client, which is generated in the TPM. This shared secret key must be marked as a non-migratable key and the public part must be transferred to the server. This key is used to securely exchange a key for the following communication. However, this public key must be transferred through a protected

second channel, otherwise the server is not sure that the key belongs to the correct communication partner.

Using AIK as Session Key

Another method is to use the public part of the AIK as encryption key to exchange the shared key for the following communication. In this case, the following traffic cannot be decrypted by the attacker, since he does not have the private portion of the AIK, which is stored in the protected storage of the TPM. This approach is for example proposed in [KS07] to ensure that transferred data is directly bound to a specific AIK. However, actual TPMs do not allow to decrypt data using the private part of the AIK. This is due to the specification that clearly forbids using the AIK as encryption key [Tru08].

Using Network Monitors

One alternative possibility to achieve a binding between the platform that issues the measurements and the challenger is to use a network monitoring agent [CHJ07]. This agent monitors all incoming and outgoing network connections and integrates a measurement of the network events into the PCRs. Thus, the challenger can determine if the network address of the platform that provided the measurements is equal to the network address of the challenger. At first glance, this approach seems to be a very elegant approach, however, it suffers due to the fact that it is not usable if the system is behind a router that uses network allocation tables (NAT). In addition, an attacker can easily modify his own network address, since network addresses do not provide any means of authenticity. It is thus very likely that the attacker is able to spoof the network address and is able to impersonate the address of the platform that provided the measurements.

4.2.5 *A Robust Integrity Reporting Protocol*

To overcome the masquerading attack, the malicious host, who is placed between the server and the collaborative host must be excluded from the following communication flow. Therefore, cryptographic keys must be used and included in the signed messages from C in the response phase. The specification of the TPM_Quote command allows us to include additional data and to sign it with the AIK. Thereby, it is possible to use the challenge response messages for the key-exchange.

One possibility is to adopt the Diffie-Hellman (DH) key exchange protocol [DH76] and to integrate it into the integrity

reporting protocol. One may also use RSA [RSA78] and generate the corresponding keys inside the TPM. Subsequently, these keys are used to exchange a shared session key which is used to encrypt the upcoming communication. Since the TPM does not provide support symmetric encryption for other applications, the CPU must be used for that purpose. The main difference between both approaches is that the DH parameters must be generated by the CPU whilst the RSA keys are generated by the TPM. Both approaches are equivalent in terms of security, since they both offer a secure way to exchange a key. Computing the RSA key within the TPM offers no advantage compared to computing the DH parameters with the CPU, since the shared communication key must be passed to the CPU anyway. In both cases the required random numbers can be computed by the secure random number generator of the TPM.

1a.	A	: create 160 bit nonce
1b.	A	: GenerateKey(K^{A}_{pub}, K^{A}_{priv})
2.	A $\rightarrow$ C	: ChallengeRequest(nonce,K^{A}_{pub})
3a.	C	: GenerateKey(K^{C}_{pub}, K^{C}_{priv})
3b.	C	: load AIK_{priv}
3c.	C	: retrieve Quote = $Sig(AIK_{priv}$, $(PCR, SHA1(nonce, K^{C}_{pub})))$
3d.	C	: get SML
3e.	C	: ComputeSessionKey(K^{AC})
4.	C $\rightarrow$ A	: ChallengeResponse(Quote, K^{C}_{pub}, SML) and $cert(AIK_{pub})$
5a.	A	: validate $cert(AIK_{pub})$
5b.	A	: validate Quote
5c.	A	: validate nonce and SML using PCR
5d.	A	: ComputeSessionKey(K^{AC})
6.	A	: create 160 bit $nonce_1$
7.	A $\rightarrow$ C	: ChallengeRequest($nonce_1$)
8.	C	: compute Response = $Enc(K^{AC}, nonce_1)$
9.	A $\rightarrow$ C	: ChallengeResponse(Response)
10.	A	: validate $nonce_1$

Figure 4.23: Enhanced integrity reporting protocol

To protect the integrity reporting protocol against masquerading attacks, we enhance it with a key agreement protocol. The modified integrity reporting protocol is shown in figure 4.23 with

the extension to use Diffie-Hellman parameters. It is essential for the protocol that both parties agree on one common generator g and one common group m. The asymmetric keys are then generated and computed as described in [DH76].

In step 1b, A generates $K^A_{pub} = g^a \bmod m$, while a is the private part of K^A. The major enhancement is that the attesting party generates a public key K^C_{pub} in step 3a, includes the generated key together with the PCRs and the nonce in the Quote message and signs the Quote with the AIK. The public key K^C_{pub} is generated using the client's CPU and is injected as external data into the TPM_Quote operation. Since the TPM_Quote command only allows to include 160 bits of external data, the package must be reduced to fit the 160 bit length by applying a hash function. The resulting fingerprint is sent together with K^C_{pub} to the verifier. Because C is running a trusted OS with a trustworthy platform configuration the private part of the key is not accessible to a potential malicious client. Finally, in step 5d the challenging party computes the shared session key K^{AC} by using its own private part and the public part of C. After the session key has been computed, A verifies whether C is also in possession of this shared key, by executing a second challenge-response authentication which is carried out in steps 6 - 10.

4.2.6 *Protocol Discussion*

The presented protocol prevents an attacker from camouflaging his malicious software configuration since all following messages are encrypted with the computed session key. It is also impossible for M to compute an own session key between him and A since his software is in a compromised state and his TPM is providing malicious platform configurations to A. Given that, the malicious host has no access to the private part of the generated session key which is stored on platform C. The generated symmetric session key K^{AC} has to be used to encrypt all following data, which can only be decrypted on the machine which possesses the session key. This session key cannot be transferred to the malicious host by the platform owner, since the extraction, e.g., by memory dump or by modifying the system software, would lead to a non-conformant system state which will be detected in the attestation phase.

It may be possible for the malicious client M to substitute K^A_{pub} with his own public key K^B_{pub} and then transfer this key to C. Since M cannot modify the response from C, A generates the shared session key K^{AC} which is later used for the encryption. The second challenge response authentication detects this substitution

since neither M nor C are in possession of the correct K^{AC} and are therefore not able to encrypt the delivered $nonce_1$. It may also be possible that M modifies the common generator g, or the common group m. The substitution of K^{A}_{pub} and the modification of g or m do not lead to a security problem, since in both cases the attacker would not be able to perform the second challenge-response authentication.

One could also create a bound keypair and use TPM_CertifyKey to prove that the corresponding private key is held in a trusted TPM. This asymmetric key is then used to exchange a shared symmetrical key between both parties. However, this approach is less performant compared to our solution since it requires three additional operations: The generation of an asymmetrical bound keypair, the signing of this keypair with TPM_CertifyKey and the signing of the symmetrical keypair with the generated bound keypair. Moreover, using CertifyKey would lead to additional signature verification procedures.

The integrity reporting protocol proposed in this section can be used as a building block for robust DRM systems. We present such a system in the following section.

4.2.7 *A Fair Digital Rights Management System*

In this section, we present a fair DRM system. As argued in Section 4.2, existing DRM systems neglect consumers' expectations. In particular, they bind legally purchased content to consumers' hardware and do not allow consumers to transfer their content to their other devices or to other users. However, 75% of consumers want to share music with their friends and families and would pay more for this kind of usage [DSVW05]. This willingness to pay would enable content providers to set up a new range of business modells. In the following, we sketch a DRM system which enables consumers to transfer their legally purchased content.

Figure 4.24 shows the use cases supported by our DRM system. The system enables a content provider to offer his digital content. As discussed in Section 4.2, selling digital content always implicates selling the associated license. Accordingly, purchasing digital content implicates purchasing the corresponding license. The DRM system checks the availability of an appropriate license each time the customer wants to use, e.g., render, digital content. A special kind of usage supported by our DRM system is transferring digital content from one user to another. In this case the according license or a part of it is transferred, as well.

Figure 4.25 depicts the architecture of our DRM system. The Figure shows two kinds of applications, i.e., a rendering applica-

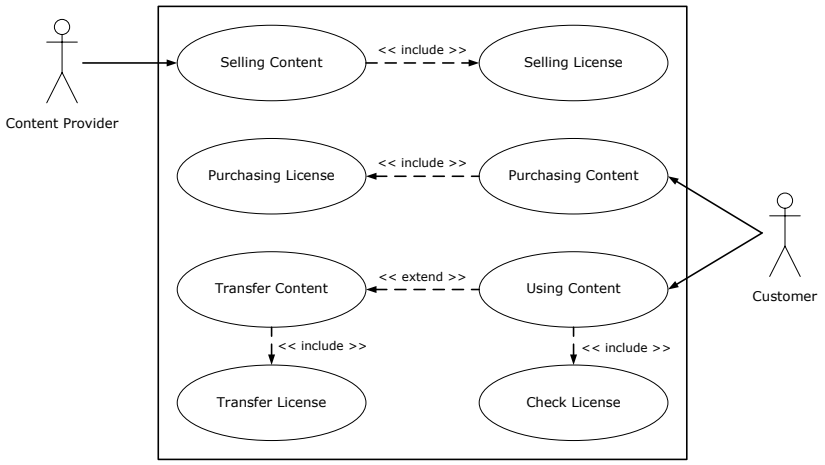

Figure 4.24: Use Case Diagram of Fair DRM System

tion on the client side and a license server. Each time a rendering application wants access to protected digital content, the DRM reference monitor is contacted which decides whether the access is granted or denied. Thus, the DRM reference monitor has similar characteristics as a policy decision point.

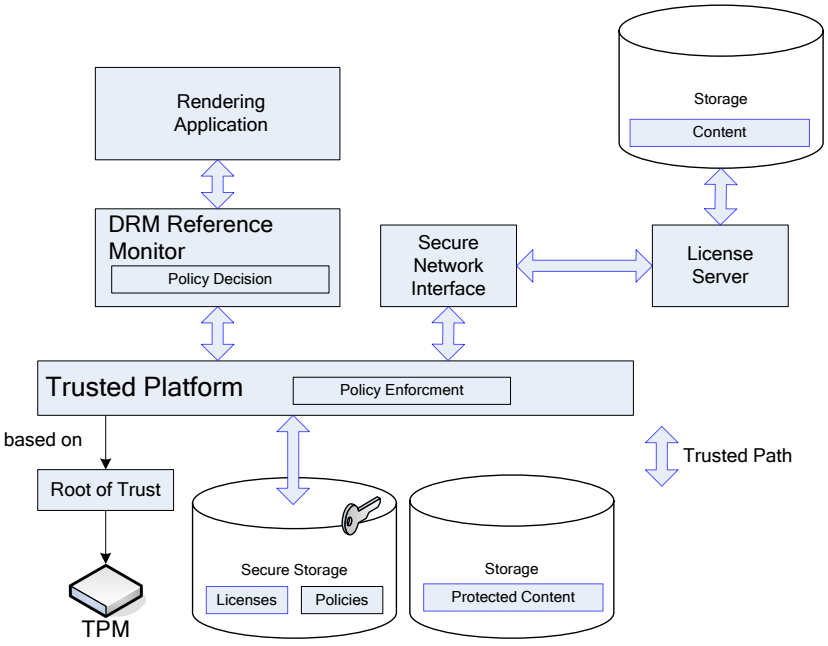

Figure 4.25: Architecture of the Fair DRM System

The DRM reference monitor itself is built on top of a trusted platform. This platform offers two functionalities, i.e., a secure storage and a secure network interface. The secure storage is necessary for storing licenses for digital content and policies, which determine the behavior of the DRM reference monitor, e.g.,

evaluating the trustworthiness of other platforms. The secure storage protects licenses and policies by sealing all data stored inside the secure storage, i.e., binding data to the state of the complete platform including the DRM reference monitor. Thus, sealed data can only be read if the DRM reference monitor is untampered and in a trusted state. We assume that access to the secure storage is only possible by the DRM reference monitor. This means that it is not possible to access the secure storage with different means rather than the interface provided to the DRM reference monitor. The secure network interface is responsible for building trusted communication channels. This is done using the robust integrity report protocol presented in Section 4.2.5.

Basically, the DRM system offers three functionalities, i.e., purchasing content, using content and transferring content. We present the necessary protocols in the following sections.

Purchasing Content

The protocol for the purchasing process is depicted in Figure 4.26. At the beginning, the content provider has to verify the trustworthiness of a client's platform, i.e., its DRM reference monitor. The integrity reporting protocol which we presented in Section 4.2.5 is used for this purpose. The result of this step is the shared session key $K_{session}$. This key is used in the next step to encrypt the customer's order including information about the digital content he wants to purchase and his intended usage. The encrypted order is transmitted to the content provider. The content provider decrypts the order with the help of $K_{session}$, generates a symmetric key, i.e., $K_{content}$ and encrypts the desired content with that key. Afterwards, he generates a license including the usage rules and $K_{content}$. The license itself is encrypted using $K_{session}$. The encrypted license and the encrypted content are sent to the customer.

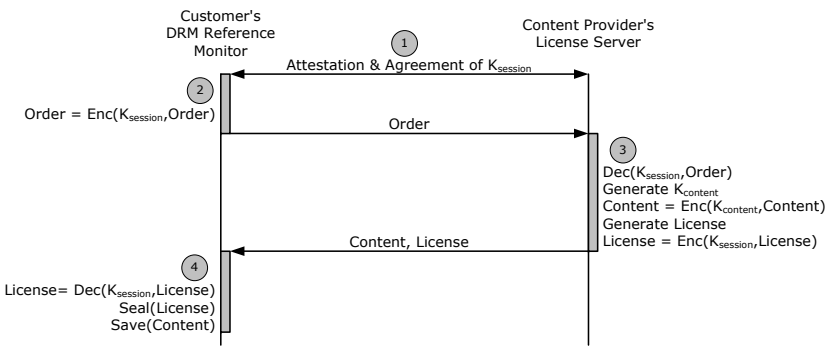

Figure 4.26: Protocol for Purchasing Digital Content

After decrypting the license, it is bound to the client's platform configuration with the help of sealing. The client saves the encrypted content without decrypting it.

Figure 4.27 shows a prototype of our DRM system. The prototype supports customers to buy content, to render content and to transfer content to other devices or users. The screenshot shows Alice's licenses. In this case, Alice is allowed to play the shown content five times and transfer it. A detailed description of the prototype is given in [Stä06].

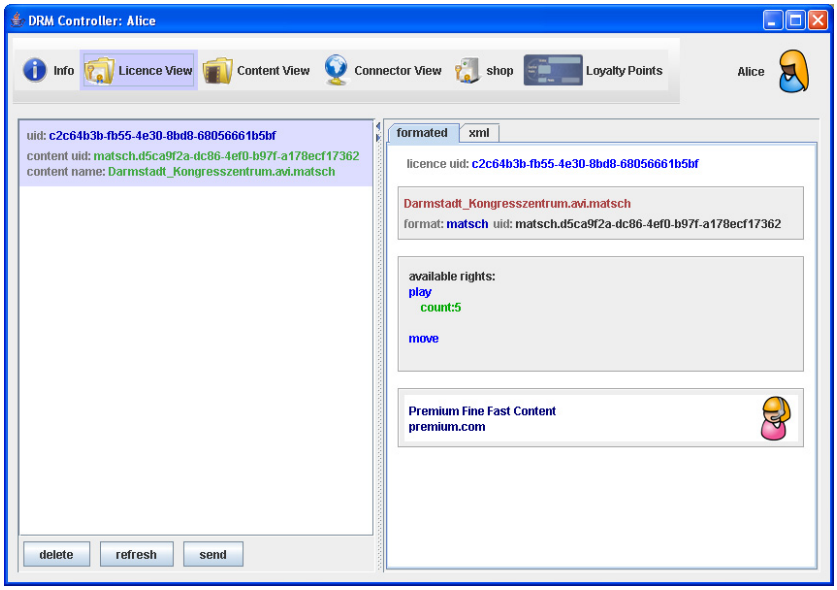

Figure 4.27: Screenshot Showing the Fair DRM System

Using Content

Figure 4.28 depicts the process of using content. A rendering application and a DRM reference monitor are involved in this process. Both applications are located at the user's platform (Compare Figure 4.25). At the beginning, the rendering application has to prove its trustworthiness with the help of attestation. Based on the received integrity information, the DRM reference monitor decides whether the rendering application is trustworthy or not. For this, we use the protocol proposed in Section 4.2.5. The resulting key $K_{session}$ is used to ensure confidentiality during the next steps. The rendering application prepares a UsageRequest. This message is encrypted with $K_{session}$ and contains information about the content and the intended usage. After receiving the UsageRequest, the DRM reference monitor decrypts it with $K_{session}$ and checks the availability of the corresponding license. If the license is available, the DRM reference monitor checks

whether the customer possesses the necessary usage rights which are given in the UsageRequest. In the case of a positive result, the DRM reference monitor decrypts the encrypted content with $K_{content}$ which is stored in the license. Afterwards, the license is encrypted again with the $K_{session}$. This approach ensures that only the trusted rendering application can access the content. Before sending the encrypted content, the DRM reference monitor updates the license. This is an optional action and is only necessary when the license limits the number of usage. After receiving the encrypted content, the rendering application decrypts it with $K_{session}$ and renders it to the user.

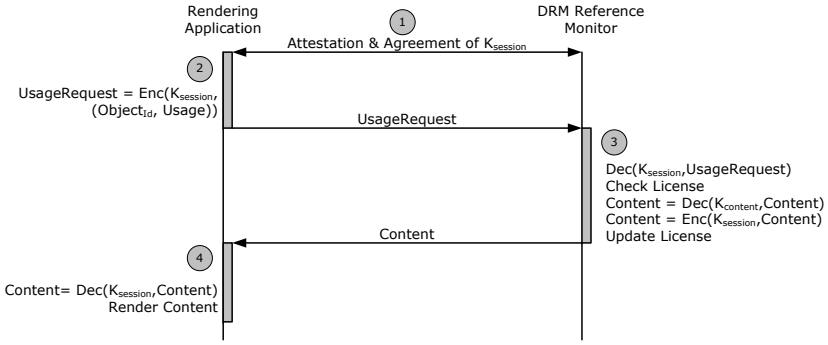

Figure 4.28: Protocol for Using Content

A critical attack on the secure storage of our DRM system are replay attacks, where a user replays the actual secure storage with an old secure storage. This attack is especially relevant if a user only possesses a limited number of usage rights on a particular protected content. To achieve protection against attacks of this type, the monotic counter of the TPM could be used as already explained in [SSSW06] or [SE08].

Transferring Content

Figure 4.29 shows the process of transferring digital content from one customer to another customer. This kind of usage is a key feature of our DRM system and does not only support the exchange of digital content between customers. In addition, it allows the transfer of digital content between devices of one customer. As argued in Section 4.2, customers value this kind of usage and would pay for it.

We assume two customers in our scenario, namely Alice and Bob. Alice wants to transfer a certain digital content or part of it to Bob. At the beginning, Alice's DRM reference monitor checks whether she has the right to transfer the content in question. This permission is stored in the corresponding license, which can be

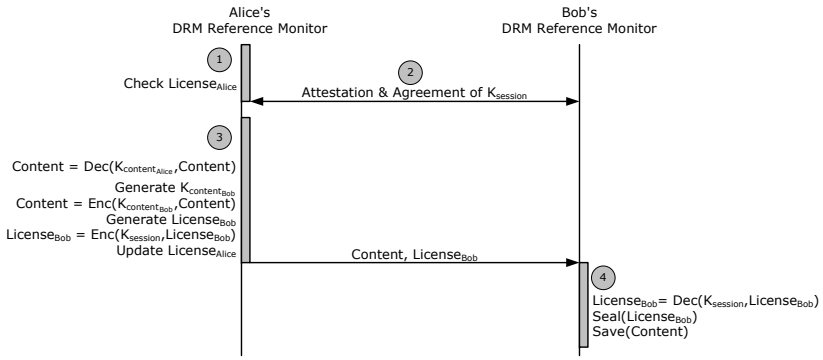

Figure 4.29: Protocol for Transferring Content

accessed only if Alice's DRM reference monitor is trusted. If Alice has the right to transfer the content, her DRM reference monitor requires an attestation from Bob's DRM reference monitor to verify its trustworthiness. For this purpose, we apply the protocol presented in Section 4.2.5. The key $K_{session}$ is the result of this step and is used for the next steps to ensure that only Bob's DRM reference monitor can access the upcoming messages. After the attestation, Alice's DRM reference monitor decrypts the content using the symmetric key $K_{content_{Alice}}$. This key is stored in the corresponding license, which is only accessible if Alice's DRM reference monitor is untampered. After this decryption, Alice's DRM reference monitor generates the symmetric key $K_{content_{Bob}}$ and encrypts the content using that key. Afterwards, Alice's DRM reference monitor generates a license for Bob and updates its own license. Bob's license includes the $K_{content_{Bob}}$ and usage rights and is encrypted with $K_{session}$ and is sent together with the encrypted content to Bob's DRM. After receiving the message, Bob's DRM reference monitor decrypts the license and binds it to Bob's platform configuration by sealing it to the actual platform configuration. Afterwards, Bob's DRM reference monitor stores the encrypted content.

The process of transferring digital content between different users or devices of one user is similar to the process of purchasing content except for two additional steps. These are checking the sender's license with respect to transfer rights and updating the sender's license before generating and transferring the receiver's license. In contrast to purchasing content, transferring content does not require any interactions with the content provider. However, the proposed protocol ensures that the transferred content can only be used according to usage rules, which were defined by the content provider at the time of purchase.

To prevent that a customer transfers digital content to another customer and then replays an old secure storage, we again pro-

pose to use the monotonic counter of the TPM. Using this concept, it can be ensured that a misbehaving customer cannot replay an old secure storage after he transferred digital content to another customer.

4.3 RELATED WORK

In the following, we review related work proposed by the scientific community and compare them with our contributions in this section.

Access Control for XML-Documents

The model proposed in [BF02] supports selective authorizations to parts of documents based on the semantic structure of XML documents. Authorizations can be defined for different nodes together with propagation options. Regarding these aspects, the model is very similar to our work. However, the supported operations and their semantics are different, since our approach is able to differentiate between objects with different histories. The support of copying data differs from our work, since the model proposed in [BF02] supports only a push of different views of a document to different sets of users, whereas our model allows us to define which elements of one document may be reused in other documents. Similar approaches can be found in [DCPS00, DdVPS02, GB02, MTK06], where [GB02, MTK06] consider access control rules for the read operation only. All these approaches consider the XML element as the smallest unit of protection, in contrast to our approach, which is capable of handling parts of the text stored in an XML element.

Iwaihara et al. allow to define access based on the version relationship of documents and elements among each other [ICAW05]. They define six operations including copy, which is similar to our copy operation, but can only be applied to elements or subtrees and not to text content or parts of the text content. In contrast to our model, the modification of the text content of an element is modeled by the operation update only, which describes that the entire content of a node is replaced with a new content. Concerning AC, Iwaihara et al. only consider read and write operations and do not define a copy operation as part of their privileges. Consequently, they cannot express which transfers among documents are permitted or denied. Moreover, they do not have the concept of splitting copied elements to have different history information for parts from different sources.

Since the specifications of the TCG are still in progress, many open issues exist. There is a large number of work focusing on the concepts of trusted computing. [SZJD04] presents a comprehensive prototype based on trusted computing technologies. In particular, it comes up with an architecture for integrity measurement, which contains the integrity reporting protocol, which we enhanced in Section 4.2.5 to make it resistant against masquerading attacks.

Terra [GPC+03] provides an approach for remote attestation. It supports the integrity measurement of virtual machines providing runtime environment for sets of processes. The approach does not build up on TPM and does not provide a protocol for integrity reporting to remote entities, which are the main differences to our work.

Our work is based on the assumption that a trusted OS provides us with the measurement of all executed code, i.e. binary attestation. In contrast to that, [HCF04, SS04] focus on semantic attestation based on attesting the behavior of software components. Nevertheless, these approaches require also robust integrity protocols for submitting their measured properties.

[BLP05] proposes the integration of key exchange protocols into Direct Anonymous Attestation (DAA) [BCC04] in P2P networks, which is basically similar to our approach. However, the objectives of the integration of key exchange protocols are different, since [BLP05] aims at building stable identities in P2P networks. Additionally, the presented approach does not feature integrity reporting and cannot be directly applied to remote attestation.

Another related work is [GPS06] which aims at building secure tunnels between endpoints. But this approach adds a new platform property certificate, which links the AIK to the TLS-Certificate. Moreover, the presented approach focuses on server attestation, which needs in turn an additional trusted CA that offers the platform property certificate. In contrast to that, our approach focuses on client attestation without an additional trusted CA, since we directly bind the cryptographic channel to the AIK.

4.4 CONCLUSIONS

In this chapter, we focused on organized business processes. In this kind of business processes, the involved parties have more stable relationships with each other. Stable relationships allow the use of hard security mechanisms, since this kind of mecha-

nisms require established settings, e.g., shared keys and known identities. In this context, we addressed usage control which is a prerequisite for a wide range of business processes. Usage control demands two functionalities. First, we need methods for defining usage rules. Second, mechanisms for enforcing the defined usage rules are required. In this section, we addressed both aspects of usage control.

In Section 4.1, we proposed a model for controlling access to XML documents. The main features of our model are its flexibility and expressiveness. Both are the result of our history-based approach for addressing objects in an access control rule. We use histories to record operations which are performed on an XML document or parts of it. The history information can be used for specifying objects in access control rules. This approach allows us to define usage rules.

In Section 4.2, we focused on enforcing defined usage rules on remote platforms. For this, we proposed a robust integrity reporting protocol which is necessary when a remote platform has to perform security relevant operations, e.g., to enforce a security policy. This protocol can be used for developing a robust Digital Rights Management (DRM) system that supports transfer of digital goods to other users or devices. We presented such a DRM system in Section 4.2.7.

5

CONCLUSIONS AND FUTURE WORK

In this work, we had been primarily concerned with trust building and usage control in context of business processes. In Section 2, we argued why addressing these two topics is crucial when companies make an effort to streamline their business processes with the help of information technology.

In Section 3, we proposed a reputation system for electronic negotiations to enable market participants to evaluate each other and their offered goods. For this purpose, we developed a system architecture which allows business partners to evaluate each other with the help of ratings after a concluded transaction. These ratings are aggregated to support a market participant to find a trustworthy business partner. This aggregation is based on the concept of Web of Trust to make the reputation system robust against malicious behavior aiming at manipulating the reputation of some market participants. In addition to the aggregated values, our approach offers differentiated views for assessing a business partner's capabilities. Instead of an overall information, a potential partner can get insight about the business behavior and the quality of the offered goods separately, since these do not necessarily depend on each other.

In addition to the central reputation system, we proposed a decentralized reputation system for P2P networks. We designed and implemented a set of protocols supporting trust building in P2P networks. Our protocols allow peers to rate each other after concluded transactions. This is done without any trusted third party. This feature preserves the decentralized character of P2P networks. Special attention was paid to the robustness of the proposed reputation system. For that purpose, we analyzed existing algorithms for rating aggregations with help of a simulation and found a shortcoming. We found out the impact of malicious behavior depends on the number of transactions in a market. In other words, the computed trust value of a malicious peer increases in course of time. To solve this issue, we proposed a robust algorithm for rating aggregation and proved its efficiency with the help of simulation.

In addition to the reputation systems, we have analyzed a concrete business process, i.e., Sale by Call for Tenders (CFT), with respect to trust building. It was shown that a naive adoption of the CFT to open market places is subject to security attacks like eavesdropping of messages, masquerading of identity, message manipulation, and repudiation of messages. The new secure CFT protocol introduced in Section 3.5.2 shows how the basic CFT can be supplemented with non-repudiable communications including authentication of origin, message integrity, and transaction authentication. This approach ensures that no new manipulation possibilities are introduced by the realization of the CFT protocol with computer support.

Our analysis revealed that further drawbacks of the CFT protocol surface in the economic design of the protocol. The main source of manipulation is caused by the unequal distribution of knowledge and of control between buyers and sellers within the CFT process. The secure sealed-bid auction protocol introduced in Section 3.5.3 provides an interesting alternative to a CFT because it overcomes these limitations. It provides means for a bidder to verify the result of the auction process, and thus, introduces trust among the participants because a correct application of the winner determination rules is guaranteed.

In Section 4, we addressed usage control from different perspectives. In Section 4.1, we showed how to define usage rules for controlling access to XML documents in a flexible way. In Section 4.2, we explained how to enforce security polices, i.e., usage rules.

In Section 4.1, we have presented a model for access control for XML documents. In our model, access to a document and its parts can be defined based both on the current document content and on the history information that captures the operations performed on that document. Moreover, the history information includes the source of the parts of a document that were transferred from different documents. Our model provides a mechanism to keep track of the parts of the text content of an XML element, where the parts were transferred from different locations or were created in different contexts. This enables us to have protection units smaller than the XML element, which leads to a finer granularity that can enhance both security and usability.

The objects in our model are defined by XPath patterns. For that purpose, we have extended XPath to access history information. Additionally, we have defined five operations which can be invoked by human subjects to view and manipulate these objects. The effects of the operations are recorded in histories.

Afterwards, we presented a system architecture that enables us to apply the model in a scenario where multiple users concurrently edit documents in an efficient way. The proposed system architecture maintains the rules, the documents and the history, so that this information is accessible for access decisions of the PDP. We introduced the check-in and check-out approach, which reduces the overhead of recalculating permissions for dependent documents. We specified the workflow for editing a document by explaining all necessary algorithms including the algorithm for the calculation of permissions and the algorithm for the creation of views.

We addressed the issue of controlling access to digital content in Section 4.2. First, we analyzed existing DRM systems with respect to customer acceptance and security. We found out, that current DRM systems rely on protection measures on the basis of unreliable software-based solutions. In addition, they do not sufficiently consider customers' expectations. To overcome these shortcomings, we proposed to use trusted computing technologies because of two reasons. First, they allow us to build hardware-based (and thus more robust) DRM systems. Second, trust computing is an emerging technology and will be available in different platforms, in particular mobile devices [Sado8]. This trend will spur the interoperability of systems based on trusted computing technologies and allow consumers to transfer and use their purchased content to other devices and/or users.

Remote attestation is one of the key technologies specified by the TCG. It offers the possibility to attest the configuration of the local platform to a remote platform with the help of integrity reporting protocols. We have shown that masquerading attacks against integrity reporting protocols are possible. This evolves because the AIK does not reveal whether the attesting system is the one, which really provides the measured values. This shortcoming may become relevant when remote attestation is used to guarantee the trustworthiness of client systems, such as DRM-clients. To address this issue, we have presented a robust protocol which prevents masquerading attacks. For that purpose, we added a key agreement scheme into an integrity reporting protocol. The resulting protocol guarantees that the communication after the remote attestation is authentic, i.e. the challenger communicates with the system that provided the measurement values.

In Section 4.2.7, we presented a fair DRM system. The previously mentioned integrity reporting protocol is a building block of the proposed DRM system. This system considers both, the requirements of content providers and the requirements of con-

sumers. It provides a high protection level. At the same time, it allows users to transfer their purchased content to other devices or users.

Finally, DRM itself is a strongly debated topic. In course of time, companies may develop alternative business models that do not require protection of copyright in its current form. However, the integrity reporting presented in Section 4.2.5 is also useful as a building block in other business processes.

FUTURE WORK

Although this thesis has shown a range of security mechanisms for trust building and usage control in context of business processes, it opens up some research opportunities and questions that are still unanswered. In the following, we describes some of these important areas.

Reputation Systems

The reputation system proposed in Section 3.3 is a central reputation system with a trusted party. Since this party can observe the complete market including all transaction and corresponding ratings, it can apply mechanisms to detect attacks against the reputation system. These mechanisms could base on clustering of market participants on the basis of their bilateral ratings and would be a reasonable complement to our reputation system which aggregates ratings from a certain market participant's point of view.

As shown in Section 3.4.8.4, transitive rating aggregation is robust against ballot stuffing and bad-mouthing. However, this approach is not applicable in case of new peers entering the market. In this case no trust paths can be found. Thus, transitive rating aggregation is useless for this kind of peers. Instead, they could use other rating aggregation mechanisms which consider only the rating list of the peer in question. Such a mechanism could analyze a rating list with respect to discrepancies which could occur as a result of malicious behavior. For instance, in case of ballot stuffing, a malicious peer receives numerous high ratings from a small group of (other malicious) peers. Given this fact, we could divide the ratees into two groups according to the number of ratings. One group contains ratees who have given several ratings whereas the other group consists of those ratees who have given only few ratings. Then, we calculate the average rating value for each group and compare the two results with

each other. We assume that a significant difference between the two average values is a sign for ballot stuffing.

The reputation systems proposed in this thesis aim at trust building towards human actors. However, since autonomous computer systems, e.g., a service, may be involved in business processes, it would be necessary to develop mechanisms to assess the trustworthiness of technical systems. This kind of mechanism could base on trusted parties or witnesses that observe transactions, e.g., using a service, and make their observations available for other interested parties.

Fair Auctions

Our protocol for sealed-bid auctions presented in Section 3.5.3 assumes a trusted third party (TTP). However, this assumption causes an overhead for setting up a TTP. Thus, developing a protocol for sealed-bid auctions which does not rely on TTPs would be a more feasible solution.

A sealed-bid auction is only one possible form of pricing mechanisms. Other pricing mechanisms need other kinds of methods for trust building. We explain this with the help of a concrete pricing mechanisms, i.e., reverse pricing. It is a dynamic pricing mechanism. The very basic idea of it is simple: a seller first specifies a secret threshold price for a certain product and offers it to potential buyers. By naming a price a potential buyer indicates his willingness to buy the product at his price. The deal is conducted if the buyer's price exceeds the seller's threshold. Otherwise the seller's offer remains open. Since the threshold price is secret, bidders have to rely on the seller's actions and decisions. In particular, a bidder has to trust that a not accepted bid means that the bid was below the seller's threshold price. In other words, reverse pricing, if solely based on trust, usually lacks transparency which may lead to unfair behavior. Thus, we need a protocol which meets two requirements. First, the seller should not be able to increase his initial threshold price while keeping it confidential. Second, a bidder should be able to verify whether his bid exceeds a seller's threshold without learning the threshold.

Access Control for XML-Documents

One possible enhancement of our model would be supporting provisions and obligations [HBP05]. These concepts define conditions, which must be fulfilled before access is granted or after access has been granted respectively, e.g, a condition stating that the user must sign an agreement before he can access certain

data. We can extend our model by adding another field to our access control rules which specify provisions and obligations.

Another open issue in this field is the efficiency analysis of our access control model. In particular, the algorithms for rule evaluation and view creation should be analyzed with respect to runtime and memory usage.

BIBLIOGRAPHY

[Ake70] G. A. Akerlof. The Market for 'Lemons': Quality Uncertainty and the Market Mechanism. *Quartely Journal of Economics*, 84(3):488–500, 1970.

[ALA01] R. Au, M. Looi, and P. Ashley. Automated Cross-Organisational Trust Establishment on Extranets. In *ITVE '01: Proceedings of the Workshop on Information Technology for Virtual Enterprises*, pages 3–11, Washington, DC, USA, 2001. IEEE Computer Society.

[App08] Apple. FairPlay DRM. http://www.apple.com/itunes/, 2008.

[ATS04] S. Androutsellis-Theotokis and D. Spinellis. A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Computing Surveys (CSUR)*, 36(4):335–371, 2004.

[Bak98] Y. Bakos. The Emerging Role of Electronic Marketplaces on the Internet. *Communincations of the ACM*, 41(8):35–42, 1998.

[BBC+07] A. Berglund, S. Boag, D. Chamberlin, M. F. Fernández, M. Kay, J. Robie, and J. Siméon. XML Path Language (XPath) Version 2.0. W3C Recommendation, World Wide Web Consortium (W3C), January 2007. http://www.w3.org/TR/xpath20/.

[BBGR03] E. Becker, W. Buhse, D. Günnewig, and N. Rump, editors. *Digital Rights Management - Technological, Economic, Legal and Political Aspects*, volume 2770 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

[BCC04] E. Brickell, J. Camenisch, and L. Chen. Direct Anonymous Attestation. In *CCS '04: Proceedings of the 11th ACM Conference on Computer and Communications Security*, pages 132–145. ACM Press, 2004.

[BCP+03] B. Balacheff, L. Chen, S. Pearson, D. Plaquin, and G. Proudler. *Trusted Computing Platforms*. Hewlett-Packard Company, 2003.

[BF02] E. Bertino and E. Ferrari. Secure and Selective Dissemination of XML Documents. *ACM Transactions on Information and System Security*, 5(3):290–331, 2002.

[Bic00] M. Bichler. A Roadmap to Auction-based Negotiation Protocols for Electronic Commerce. In *HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences*, volume 6, Washington, DC, USA, 2000. IEEE Computer Society.

[BL73] D. Bell and L. LaPadula. Secure Computer Systems: Mathematical Foundations and Model. Technical Report M74-244, MITRE Corp, Bedfort, MA, 1973.

[BLP05] S. Balfe, A. D. Lakhani, and K. G. Paterson. Trusted Computing: Providing Security for Peer-to-Peer Networks. In *P2P '05: Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing (P2P'05)*, pages 117–124, Washington, DC, USA, 2005. IEEE Computer Society.

[BM92] S. M. Bellovin and M. Merritt. Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks. In *Proceedings ot IEEE Symposium on Research in Security and Privacy*, pages 72–84, 1992.

[BN89] F. D. Brewer and J. M. Nash. The Chinese Wall Security Policy. In *IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 1989.

[BS97] C. Beam and A. Segev. Automated Negotiations: A Survey of the State of the Art. *Wirtschaftsinformatik*, 39(3):263–268, 1997.

[Cac99] C. Cachin. Efficient Private Bidding and Auctions with an Oblivious Third Party. In *Proceedings of the 5th ACM Conference on Computer and Communications Security*, pages 120–127, 1999.

[Car00] G. Caronni. Walking the Web of Trust. In *WETICE '00: Proceedings of the 9th IEEE International Workshops on Enabling Technologies*, pages 153–158, Washington, DC, USA, 2000. IEEE Computer Society.

[CGLZ03] K. Campbell, L. A. Gordon, M. P. Loeb, and L. Zhou. The economic cost of publicly announced information security breaches: empirical evidence from the

stock market. *Journal of Computer Security*, 11(3):431–448, 2003.

[CH07] E. Carrara and G. Hogben. Reputation-based systems: a security analysis. ENISA Position Paper, October 2007.

[Cha85] D. Chaum. Security without Identification: Transaction Systems to Make Big Brother Obsolete. *Communications of the ACM*, 28(10):1030–1044, 1985.

[Cha88] D. Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of Cryptology*, I(1), 1988.

[CHJ07] S. Choi, J. Han, and S. Jun. Improvement on TCG Attestation and Its Implication for DRM. In *Proceedings of Computational Science and Its Applications - ICCSA 2007*, volume 4705 of *Lecture Notes in Computer Science*, pages 912–925. Springer-Verlag, August 2007.

[Con09] The MultiNeg Consortium. http:www.fbw.h-da.de/multineg, 2009.

[CT01] C. Castelfranchi and Y. Tan. The Role of Trust and Deception in Virtual Societies. In *Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 7*, volume 7. IEEE Computer Society, 2001.

[Dav92] T. H. Davenport. *Process Innovation: Reengineering Work Through Information Technology*. Harvard Business School Press, 1992.

[DCPS00] E. Damiani, S. De Capitani, S. Paraboschi, and P. Samarati. Securing XML Documents. In *Proceedings of the 7th International Conference on Extending Database Technology*, volume 1777 of *Lecture Notes in Computer Science*, pages 121–135. Springer-Verlag, 2000.

[DdVPS02] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. A Fine-Grained Access Control System for XML Documents. *ACM Transactions on Information and System Security (TISSEC)*, 5(2):169–202, 2002.

[Del00] C. Dellarocas. Immunizing Online Reputation Reporting Systems Against Unfair Ratings and Discriminatory Behavior. In *EC '00: Proceedings of the*

2nd ACM conference on Electronic commerce, pages 150–157. ACM Press, 2000.

[Del01] C. Dellarocas. Analyzing the Economic Efficiency of eBay-like Online Reputation Reporting Mechanisms. In *EC '01: Proceedings of the 3rd ACM conference on Electronic Commerce*, pages 171–179. ACM Press, 2001.

[DFF+07] D. Draper, P. Fankhauser, M. Fernández, A. Malhotra, K. Rose, M. Rys, J. Siméon, and P. Wadler. XQuery 1.0 and XPath 2.0 Formal Semantics. Technical report, World Wide Web Consortium (W3C), January 2007. http://www.w3.org/TR/xquery-semantics/.

[DH76] W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[Dou02] J. R. Douceur. The Sybil Attack. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, volume 2429 of *Lecture Notes in Computer Science*, pages 251–260. Springer-Verlag, 2002.

[DR06] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346, April 2006.

[DSVW05] N. Dufft, A. Stiehler, D. Vogeley, and T. Wichmann. Digital Music Usage and DRM - Results from an European Consumer Survey, May 2005.

[ER02] M. Ettredge and V. Richardson. Assessing the Risk in E-commerce. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)-Volume 7*, page 194. IEEE Computer Society, 2002.

[FC05] M. Feldman and J. Chuang. Overcoming Free-Riding Behavior in Peer-to-Peer Systems. *ACM SIGecom Exchanges*, 5(4):41–50, 2005.

[Fet03] M. Fetscherin. Evaluating Consumer Acceptance for Protected Digital Content. In Becker et al. [BBGR03], pages 321–333.

[FR96] M. K. Franklin and M. K. Reiter. The Design and Implementation of a Secure Auction Service. *IEEE*

Transactions on Software Engineering, 22(5):302–312, 1996.

[Gar06] Gartner. New Gartner Hype Cycle Highlights Five High Impact IT Security Risks. http://www.gartner.com/it/page.jsp?id=496247, 2006.

[GB02] A. Gabillon and E. Bruno. Regulating Access to XML Documents. In *Working Conference on Database and Application Security*, pages 299–314. Kluwer Academic Publishers, 2002.

[GD72] G. S. Graham and P. J. Denning. Protection - Principles and Practice. In *Spring Joint Computer Reference*, volume 40, pages 417–429, 1972.

[GJA03] M. Gupta, P. Judge, and M. Ammar. A Reputation System for Peer-to-Peer Networks. In *NOSSDAV '03: Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*, pages 144–152. ACM Press, 2003.

[GKS04] A. Gadatsch, T. Knuppertz, and S. Schnägelberger. Geschäftsprozessmanagement - Eine Umfrage zur aktuellen Situation in Deutschland. Schriftenreihe des Fachbereiches Wirtschaft Sankt Augustin, 2004.

[GLLR06] L. A. Gordon, M. P. Loeb, W. Lucyshyn, and R. Richardson. 2006 CSI/FBI Computer Crime and Security Survey. Technical report, CSI, 2006.

[GM82] J. A. Goguen and J. Meseguer. Security Policies and Security Models. In *IEEE Symposium on Security and Privacy*, pages 11–20, 1982.

[GPC+03] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: A Virtual Machine-Based Platform for Trusted Computing. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating Systems Principles*, pages 193–206. ACM Press, 2003.

[GPS06] K. Goldman, R. Perez, and R. Sailer. Linking Remote Attestation to Secure Tunnel Endpoints. In *First ACM Workshop on Scalable Trusted Computing*, Fairfax, Virginia, November 2006.

[GRS96] D. Goldschlag, M. Reed, and P. Syverson. Hiding routing information. In *Information Hiding*, volume

1174 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996.

[Gut03] Susanne Guth. Rights expression languages. In Becker et al. [BBGR03], pages 101–112.

[Han06] M. Hansen. DRM-Desaster: Das Sony BMG-Rootkit. *Datenschutz und Datensicherheit (DuD)*, 30(2):95–97, 2006.

[HBP05] M. Hilty, David A. Basin, and Alexander Pretschner. On Obligations. In *Proceedings of 10th European Symposium on Research in Computer Security (ESORICS 2005)*, volume 3679 of *Lecture Notes in Computer Science*, pages 98–117. Springer-Verlag, 2005.

[HC93] M. Hammer and J. Champy. *Reengineering the Corporation: A Manifesto for Business Revolution*. HarperBusiness, 1993.

[HCF04] V. Haldar, D. Chandra, and M. Franz. Semantic Remote attestation: A Virtual Machine directed approach to Trusted Computing. In *USENIX Virtual Machine Research and Technology Symposium*, 2004 2004.

[HKT98] M. Harkavy, H. Kikuchi, and J. D. Tygar. Electronic Auctions with Private Bids. In *Third USENIX Workshop on Electronic Commerce*, Boston, USA, September 1998.

[Ian01] R. Iannella. Digital Rights Management (DRM) Architectures. *D-Lib Magazine*, 7(6), 2001.

[ICAW05] Mizuho Iwaihara, Somchai Chatvichienchai, Chutiporn Anutariya, and Vilas Wuwongse. Relevancy Based Access Control of Versioned XML Documents. In *SACMAT '05: Proceedings of the tenth ACM Symposium on Access Control Models and Technologies*, pages 85–94. ACM Press, 2005.

[Ins07] Computer Security Institute. 2007 CSI Computer Crime and Security Survey, 2007.

[JHP06] A. Jøsang, R. Hayward, and S. Pope. Trust Network Analysis with Subjective Logic. In *ACSC '06: Proceedings of the 29th Australasian Computer Science Conference*, pages 85–94, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc.

[JIB07] A. Jøsang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618 – 644, 2007.

[KF98a] M. Kumar and S. Feldman. Internet Auctions. In *Proceedings of the 3rd USENIX Workshop on Electronic Commerce*, aug 1998.

[KF98b] M. Kumar and S. I. Feldman. Business negotiations on the Internet. IAC Reports, IBM T.J. Watson Research Center, March 1998.

[KF98c] M. Kumar and S. I. Feldman. Internet Auctions. In *Proceedings of the 3rd conference on USENIX Workshop on Electronic Commerce (WOEC'98)*, November 1998.

[KG03] D. Kuhlmann and R. Gehring. Trusted Platforms, DRM, and Beyond. In Becker et al. [BBGR03], pages 178–205.

[KHT98] H. Kikuchi, M. Harkavy, and J. Tygar. Multi-round Anonymous Auction Protocols. In *Proceedings of the first IEEE Workshop on Dependable and Real-Time E-Commerce Systems*, jun 1998.

[Kle00] S. Klein. *The Emergence of Auctions on the World Wide Web*. Springer-Verlag, 2000.

[KR00] R. Kalakota and M. Robinson. *E-Business 2.0: Roadmap for Success*. Addison-Wesley Professional, 2000.

[KS07] N. Kuntze and A. U. Schmidt. Protection of DVB Systems by Trusted Computing. In *Proceedings of the IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*, 2007.

[KSGM03] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The Eigentrust Algorithm for Reputation Management in P2P Networks. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 640–651. ACM Press, 2003.

[LDL03] A. Lindsay, D. Downs, and K. Lunn. Business processes–attempts to find a definition. *Information & Software Technology*, 45(15):1015–1019, 2003.

[LR04] S. K. Lam and J. Riedl. Shilling Recommender Systems for Fun and Profit. In *Proceedings of the 13th international conference on World Wide Web (WWW '04)*, pages 393–402. ACM, 2004.

[LSNS03] Q. Liu, R. Safavi-Naini, and N. P. Sheppard. Digital Rights Management for Content Distribution. In *ACSW Frontiers '03: Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003*, pages 49–58, Darlinghurst, Australia, Australia, 2003. Australian Computer Society, Inc.

[LZRD04] L. Liu, S. Zhang, K. D. Ryu, and P. Dasgupta. R-Chain: A Self-Maintained Reputation Management System in P2P Networks. In *Proceedings of the ISCA 17th International Conference on Parallel and Distributed Computing Systems*, pages 131–136, 2004.

[LZY05] P. Liu, W. Zang, and M. Yu. Incentive-Based Modeling and Inference of Attacker Intent, Objectives, and Strategies. *ACM Transactions on Information and System Security*, 8(1):78–118, February 2005.

[Man03] H. Mantel. *A Uniform Framework for the Formal Specification and Verification of Information Flow Security*. PhD thesis, Universität des Saarlandes, 2003.

[Mau96] U. M. Maurer. Modelling a Public-Key Infrastructure. In *4th European Symposium on Research in Computer Security (ESORICS 96)*, volume 1146 of *Lecture Notes in Computer Science*, pages 325–350. Springer-Verlag, 1996.

[MGM06] S. Marti and H. Garcia-Molina. Taxonomy of trust: Categorizing P2P reputation systems. *Computer Networks*, 50(4):472–484, 2006.

[Mic08] Microsoft. Windows media rights manager 10. http://www.microsoft.com/windows/windowsmedia/drm/default.aspx, 2008.

[MKL+02] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu. Peer-to-Peer Computing. Technical Report HPL-2002-57 (R.1), HP Laboratories Palo Alto, 2002.

[MM87] R. P. McAfee and J. McMillan. Auctions and bidding. *Journal of Economic Literature*, XXV:699–738, 1987.

[MS96] A. Mårtensson and G. Steneskog. Business Process Excellence - Some Characteristics. In M. Lundenberg and B. Sundgren, editors, *Advancing your Business: People and Information Systrems in Concert*. EFI, Stockholm University of Economics, 1996.

[MTK06] M. Murata, A. Tozawa, and M. Kudo. XML Access Control using Static Analysis. *ACM Transactions on Information and System Security*, 9(3):292–324, 2006.

[Mui02] L. Mui. *Computational Models of Trust and Reputation: Agents, Evolutionary Games, and Social Networks*. PhD thesis, Massachusetts Institute of Technology, December 2002.

[NPS99] M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the first ACM Conference on Electronic Commerce*, nov 1999.

[Rö8] P. Röder. *History-Based Access Control for XML Documents*. PhD thesis, TU Darmstadt, 2008.

[RBR+04] M. Roussopoulos, M. Baker, D. S. H. Rosenthal, T. J. Giuli, P. Maniatis, and J. C. Mogul. 2 P2P or Not 2 P2P? In *Peer-to-Peer Systems III, Third International Workshop, IPTPS 2004*, volume 3279 of *Lecture Notes in Computer Science*, pages 33–43. Springer-Verlag, 2004.

[RC05] J. Reid and W. Caelli. DRM, Trusted Computing and Operating System Architecture. In *Proceedings of the Australasian Information Security Workshop 2005 (AISW2005)*, pages 127–136. Australian Computer Society, Inc., 2005.

[Rip98] T. Ripperger. *Ökonomik des Vertrauens – Analyse eines Organisationsprinzips*. Mohr Siebeck, 1998.

[RJ96] L. Rasmusson and S. Jansson. Simulated Social Control for Secure Internet Commerce. In *Proceedings of the 1996 workshop on New security paradigms*, pages 18–25. ACM Press, 1996.

[RKZF00] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman. Reputation Systems. *Communications of the ACM*, 43(12):45–48, 2000.

[RL03] M. Rebstock and M. Lipp. Webservices zur Integration interaktiver elektronischer Verhandlungen in elektronische Marktplätze. *Wirtschaftsinformatik*, 45(3):293–306, 2003.

[RMAW99] W. M. Row, Donald. J. Morton, B. L. Adams, and A. H. Wright. Security issues in small linux net-

works. In *SAC '99: Proceedings of the 1999 ACM symposium on Applied computing*, pages 506–510. ACM Press, 1999.

[RMT01] W. Rosenblatt, S. Mooney, and W. Trippe. *Digital Rights Management: Business and Technology*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[RSA78] Ron L. Rivest, Adi Shamir, and Leonard M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[RSG98] M. Reed, P. Syverson, and D. Goldschlag. Anonymous Connections and Onion Routing. *IEEE Journal on Selected Areas in Communications*, 16(4), 1998.

[RT02] M. Rebstock and O. Tafreschi. Secure Interactive Electronic Negotiations In Business-to-business Marketplaces. In *Proceedings of the 10th European Conference on Information Systems, Information Systems and the Future of the Digital Economy, ECIS 2002*, June 2002.

[RT08] P. Röder and O. Tafreschi. Access Control for E-Business Integration. In M. Rebstock, J. Fengel, and H. Paulheim, editors, *Ontologies-Based Business Integration*, chapter 9, pages 173–191. Springer-Verlag, 2008.

[RTE07a] P. Röder, O. Tafreschi, and C. Eckert. History-Based Access Control for XML Documents. In *Proceedings of the ACM Symposium on Information, Computer and Communications Security (ASIACCS'07)*, pages 386–388. ACM Press, 2007.

[RTE07b] P. Röder, O. Tafreschi, and C. Eckert. On Flexible Modeling of History-Based Access Control Policies for XML Documents. In *Proceedings of the 11th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES2007:)*, volume 4694 of *Lecture Notes in Computer Science*, pages 1090–1097. Springer-Verlag, 2007.

[RTME06] P. Röder, O. Tafreschi, C. Müller, and C. Eckert. History-based Access Control and Information Flow Control for Structured Documents. In *Proceedings of the First International Workshop on Security (IWSEC 2006)*, 2006.

[RTME07] P. Röder, O. Tafreschi, F. Mellgren, and C. Eckert. A System Architecture for History-Based Access Control for XML Documents. In *Proceedings of 9th International Conference on Information and Communications Security (ICICS 2007)*, volume 4861 of *Lecture Notes in Computer Science*. Springer-Verlag, 2007.

[RTT03] M. Rebstock, P. Thun, and O. Tafreschi. Supporting Interactive Multi-Attribute Electronic Negotiations with ebXML. *Journal for Group Decision and Negotiation*, 12(4):269–286, July 2003.

[Rum03] N. Rump. Definition, Aspects, and Overview. In Becker et al. [BBGR03], pages 3–15.

[Rus05] M. Russinovich. Sony, Rootkits and Digital Rights Management Gone Too Far, 2005.

[RZ02] P. Resnick and R. Zeckhauser. Trust Among Strangers in Internet Transactions: Empirical Analysis of eBay's Reputation System. *Advances in Applied Microeconomics "The Economics of the Internet and E-Commerce"*, 11(11):127–158, 2002.

[SA99] F. Stajano and R. Anderson. The Cocaine Auction Protocol: On the Power of Anonymous Broadcast. In *Proceedings of Financial Cryptography 99*, volume 1768 of *Lecture Notes in Computer Science*, pages 434–447. Springer-Verlag, 1999.

[Sad08] A.-R. Sadeghi. Trusted Computing - Special Aspects and Challenges. In *Proceedings of SOFSEM 2008: Theory and Practice of Computer Science, 34th Conference on Current Trends in Theory and Practice of Computer Science*, volume 4910 of *Lecture Notes in Computer Science*, pages 98–117. Springer-Verlag, 2008.

[San96] T. W. Sandholm. Limitations of the Vickrey Auction in Computational Multiagent Systems. In *Proceedings of the Second International Conference on Multiagent Systems (ICMAS-96)*, pages 299–306, 1996.

[SCFY96] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-Based Access Control Models. *IEEE Computer*, 29(2):38–47, 1996.

[SE08] F. Stumpf and C. Eckert. Enhancing Trusted Platform Modules with Hardware-Based Virtualization Techniques. In *Proceedings of the Second International*

Conference on Emerging Security Information, Systems and Technologies (SECURWARE 2008). IEEE Computer Society, August 2008.

[SF03] D. Schoder and K. Fischbach. Peer-to-Peer Prospects. *Communications of the ACM*, 46(2):27–29, 2003.

[SH05] M. P. Singh and M. N. Huhns. *Service-Oriented Computing Semantics, Processes, Agents*. John Wiley & Sons, Ltd., 2005.

[SJZvD04] R. Sailer, T. Jaeger, X. Zhang, and L. van Doorn. Attestation-based Policy Enforcement for Remote Access. In *Proceedings of the 11th ACM conference on Computer and communications security (CCS '04)*, pages 308–317. ACM Press, 2004.

[SR01] L. Seligman and A. Roenthal. XML's Impact an Databases and Data Sharing. *Computer*, 34(6):59–67, 2001.

[SR06] D. Stutzbach and R. Rejaie. Understanding Churn in Peer-to-Peer Networks. In *IMC '06: Proceedings of the 6th ACM SIGCOMM on Internet measurement*, pages 189–202. ACM Press, 2006.

[SS99] S. G. Stubblebine and P. Syverson. Fair On-Line Auctions without Special Trusted Parties. In *Proceedings of Financial Cryptography 99*, volume 1648 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.

[SS04] A.-R. Sadeghi and C. Stüble. Property-based Attestation for Computing Platforms: caring about properties, not mechanisms. In *Proceedings of the 2004 workshop on New security paradigms (NSPW '04)*, pages 67–77. ACM Press, 2004.

[SSSW06] A.-R. Sadeghi, M. Scheibel, C. Stüble, and M. Wolf. Play it once again, Sam - Enforcing Stateful Licenses on Open Platforms. In *Proceedings of the Second Workshop on Advances in Trusted Computing (WATC '06)*, 2006.

[Stä06] B. Stärz. Technische Infrastrukturen zur Umsetzung neuer Erlösmodelle für digitale Güter. Diplomarbeit, TU Darmstadt, 2006.

[STRE06] F. Stumpf, O. Tafreschi, P. Röder, and C. Eckert. A Robust Integrity Reporting Protocol for Remote Attestation. In *Second Workshop on Advances in Trusted Computing (WATC'06 Fall)*, November 2006.

[STW04] A. U. Schmidt, O. Tafreschi, and R. Wolf. Interoperability Challenges for DRM Systems. In *Proceedings of the IFIP/GI Workshop on Virtual Goods*, pages 125–136, 2004.

[SVB05] A. Schlosser, M. Voss, and L. Brückner. On the Simulation of Global Reputation Systems. *Journal of Artificial Societies and Social Simulation*, 9(1), 2005.

[SvDW04] R. Sailer, L. van Doorn, and J. Ward. The Role of TPM in Enterprise Security. *Datenschutz und Datensicherheit (DuD)*, 28(9):539–544, September 2004.

[SW05] R. Steinmetz and K. Wehrle. What Is This "Peer-to-Peer" About? In *Peer-to-Peer Systems and Applications*, volume 3485 of *Lecture Notes in Computer Science*, pages 9–16. Springer-Verlag, 2005.

[SZJD04] Reiner Sailer, Xiaolan Zhang, Trent Jaeger, and Leendert van Doorn. Design and Implementation of a TCG-based Integrity Measurement Architecture. In *13th USENIX Security Symposium*. IBM T. J. Watson Research Center, August 2004.

[SZJvD04] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn. Design and Implementation of a TCG-based Integrity Measurement Architecture. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, pages 16–16. USENIX Association, 2004.

[Szt99] P. Sztompka. *Trust: A Sociological Theory (Cambridge Cultural Social Studies)*. Cambridge University Press, 1999.

[Taf07] O. Tafreschi. A Reputation System for P2P-Networks. Diplomarbeit, TU Darmstadt, 2007.

[TFR05] O. Tafreschi, J. Fengel, and M. Rebstock. Identity Management for Electronic Negotiations. In *Computer Supported Activity Coordination, Proceedings of the 2nd International Workshop on Computer Supported Activity Coordination, (CSAC 2005)*. INSTICC Press, May 2005.

[Tic85] W. F. Tichy. RCS - A System for Version Control. *Software - Practice and Experience*, 15(7):637–654, 1985.

[TMF+07] O. Tafreschi, D. Maehler, J. Fengel, M. Rebstock, and C. Eckert. A Reputation System for Electronic Negotiations. In *Proceedings of the 5th International Workshop on Security in Information Systems, WOSIS 2007*, pages 53–62. INSTICC Press, 2007.

[TMF+08] O. Tafreschi, D. Maehler, J. Fengel, M. Rebstock, and C. Eckert. A Reputation System for Electronic Negotiations. *Computer Standards & Interfaces*, 30(6):351–360, 2008.

[Tru08] Trusted Computing Group. Trusted Platform Module (TPM) specifications. Technical report, https://www.trustedcomputinggroup.org/specs/TPM, 2008.

[TSF+01] O. Tafreschi, M. Schneider, P. Fankhauser, B. Mahleko, and T. Tesch. From Call for Tenders to Sealed-Bid Auction for Mediated Ecommerce. In *Proceedings of the IFIP TC2/WG2.6 Ninth Working Conference on Database Semantics (DS-9)*, pages 69–86. Kluwer, B.V., 2001.

[TT00] Y.-H. Tan and W. Thoen. Formal Aspects of a Generic Model of Trust for Electronic Commerce. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*, volume 6. IEEE Computer Society, 2000.

[TTL00] D. Tapscott, D. Ticoll, and A. Lowy. *Digital Capital: Harnessing the Power of Business Webs*. Harvard Business School Press, 2000.

[Tur97] E. Turban. Auctions and Bidding on the Internet: An Assessment. *Electronic Markets*, 7(4), 1997.

[Tyg98] J. D. Tygar. Atomicity versus Anonymity: Distributed Transactions for Electronic Commerce. In *Proceedings of 24rd International Conference on Very Large Data Bases*, pages 1–12. Morgan Kaufmann, 1998.

[Vic61] W. Vickrey. Counter Specification, Auctions, and Competitive Sealed Tenders. *The Journal of Finance*, pages 9–37, 1961.

[VTT08] K. Vergidis, C.J. Turner, and A. Tiwari. Business process perspectives: Theoretical developments vs. real-world practice. *International Journal of Production Economics*, 114(1):91 – 104, 2008.

[VW00] P. Völkner and B. Werners. A decision support system for business process planning. *European Journal of Operational Research*, 125(3):633 – 647, 2000.

[Wal03] D. S. Wallach. A Survey of Peer-to-Peer Security Issues. In *Proceedings of the Software Security – Theories and Systems, Mext-NSF-JSPS International Symposium (ISSS 2002)*, volume 2609 of *Lecture Notes in Computer Science*, pages 42–57. Springer-Verlag, 2003.

[Wil85] R. Wilson. Reputation in Games and Markets. In A. Roth, editor, *Game-Theoretic Models of Bargaining*, pages 27–62. Cambridge University Press, 1985.

[WTL00] E. Wolff, M.T. Tu, and W. Lamersdorf. Using Genetic Algorithms to Enable Automated Auctions. In *Proceedings of the First International Conference on Electronic Commerce and Web Technologies (EC-Web 2000)*, volume 1875 of *Lecture Notes in Computer Science*, pages 389–398. Springer-Verlag, 2000.

[WvdHD98] H. Weigand, W-J. van den Heuvel, and F. Dignum. Modelling Electronic Commerce Transactions: A Layered Approach. In *Proceedings of the Third International Workshop on Communication Modelling*, pages 47–58, 1998.

[ZH06] R. Zhou and K. Hwang. PowerTrust: A Robust and Scalable Reputation System for TrustedP2P Computing. *IEEE Transactions on Parallel and Distributed Systems*, to appear, 2006.

[Zim95] P. R. Zimmermann. *The official PGP User's Guide*. MIT Press, Cambridge, MA, USA, 1995.

[ZLZ05] W. Zhang, L. Liu, and Y. Zhu. A Computational Trust Model for C2C Auctions. In *Services Systems and Services Management, Proceedings of ICSSSM '05. 2005 International Conference*, 2005.

[ZMM99] G. Zacharia, A. Moukas, and P. Maes. Collaborative Reputation Mechanisms in Electronic Marketplaces. In *Proceedings of the Thirty-second Annual Hawaii International Conference on System Sciences-Volume*, volume 8. IEEE Computer Society, 1999.